

Pse

SOR

SSSSSSSSSSSS	000000000	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	3333333333	222222222
SSSSSSSSSSSS	000000000	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	3333333333	222222222
SSSSSSSSSSSS	000000000	RRRRRRRRRRRR	TTTTTTTTTTTTTTT	3333333333	222222222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSS	000	000	RRR RRR	TTT	333 333 222 222
SSSSSSSSSS	000	000	RRRRRRRRRRRR	TTT	333 222
SSSSSSSSSS	000	000	RRRRRRRRRRRR	TTT	333 222
SSSSSSSSSS	000	000	RRRRRRRRRRRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSS	000	000	RRR RRR	TTT	333 222
SSSSSSSSSS	000000000	RRR RRR	TTT	3333333333	22222222222222
SSSSSSSSSS	000000000	RRR RRR	TTT	3333333333	22222222222222
SSSSSSSSSS	000000000	RRR RRR	TTT	3333333333	22222222222222

SOR

SOR

SOR

-LI

FILEID**SORCOLUTI

N 10

```
1      0001 0 MODULE COLL$UTILITIES(  
2          0           IDENT = 'V04-0uu'          ! File: SORCOLUTI.B32 Edit: PDG3014  
3          0           )=  
4      0004 1 BEGIN  
5          1  
6      0006 1 *****  
7      0007 1 *  
8      0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
9      0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
10     0010 1 * ALL RIGHTS RESERVED.  
11     0011 1 *  
12     0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
13     0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
14     0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
15     0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
16     0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
17     0017 1 * TRANSFERRED.  
18     0018 1 *  
19     0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
20     0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
21     0021 1 * CORPORATION.  
22     0022 1 *  
23     0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
24     0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
25     0025 1 *  
26     0026 1 *  
27     0027 1 *****  
28     0028 1 .  
29     0029 1 .  
30     0030 1 ++  
31     0031 1  
32     0032 1 FACILITY: VAX-11 SORT/MERGE  
33     0033 1 PDP-11 SORT/MERGE  
34     0034 1  
35     0035 1 ABSTRACT:  
36     0036 1  
37     0037 1 This module contains routines that process a user-defined collating  
38     0038 1 sequence.  
39     0039 1  
40     0040 1 ENVIRONMENT: VAX/VMS user mode  
41     0041 1  
42     0042 1 AUTHOR: Peter D Gilbert, CREATION DATE: 20-Jan-1983  
43     0043 1  
44     0044 1 MODIFIED BY:  
45     0045 1  
46     0046 1 T03-001 Original  
47     0047 1 T03-002 Add a temporary fix to get a reasonable pad character if  
48     0048 1 the pad character is ignored. PDG 26-Jan-1983  
49     0049 1 T03-003 Support ignored pad characters. Set ADJ to zero if there are  
50     0050 1 ignored characters. PDG 28-Jan-1983  
51     0051 1 T03-004 Add COLL$FOLD. PDG 31-Jan-1983  
52     0052 1 T03-005 Define CODE and PLIT psects. 1-Feb-1983  
53     0053 1 T03-006 Change the interface to SOR$$COLLATE x. PDG 7-Mar-1983  
54     0054 1 T03-007 Remove STATIC table stuff. Changes for PDP-11 compatibility.  
55     0055 1 PDG 5-Apr-1983  
56     0056 1 T03-008 Changes to simplify zapping the upper table. PDG 12-Apr-1983  
57     0057 1 T03-009 Store info in RES_REVERSE (not CS_REVERSE) in COLL$RESULT.
```

: 58 0058 1 :
: 59 0059 1 :
: 60 0060 1 :
: 61 0061 1 :
: 62 0062 1 :
: 63 0063 1 :
: 64 0064 1 :
: 65 0065 1 :
: 66 0066 1 :
: 67 0067 1 :
: 68 0068 1 :--

The ORDER parameter to TIE_BREAK is now required. Changed
value of CS_K_REG for Bliss-11. PDG 15-Apr-1983
T03-010 Defined error statuses for Bliss-11. PDG 21-Apr-1983
T03-011 Add routine headers. PDG 5-Jul-1983
T03-012 Allocate large structures in the work area, not on the stack.
PDG 25-Apr-1983
T03-013 Merge changes from Sort-11 and Sort-32 versions. 19-Sep-1983
T03-014 Allocate on the stack for Sort-32. Specify a comparison
routine that can be used after an initial CMPC for Sort-32
PDG 14-Oct-1983

```
: 70      0069 1 ++  
: 71      0070 1  
: 72      0071 1 OVERVIEW:  
: 73      0072 1  
: 74      0073 1 The routines must be called in the following order:  
: 75      0074 1 INIT [ BASE ] [ NEXT : MODIFY : FOLD ]... RESULT  
: 76      0075 1 The routines PAD, TIE_BREAK and UPPER may be optionally called any  
: 77      0076 1 time after the INIT and before the RESULT.  
: 78      0077 1  
: 79      0078 1 In all of these routines, the user passes a two element vector  
: 80      0079 1 containing the length/address of a work area these routines can use.  
: 81      0080 1 The call to RESULT returns the length that is needed to store the  
: 82      0081 1 compressed version of the area. The user can then call the routine  
: 83      0082 1 whose address is stored at the beginning of the area. This routine is  
: 84      0083 1 passed the lengths/address of the strings, and returns:  
: 85      0084 1     -1 if String1 < String2  
: 86      0085 1     0   if String1 = String2  
: 87      0086 1     +1 if String1 > String2  
: 88      0087 1  
: 89      0088 1 All characters are passed to these routines as a word length followed  
: 90      0089 1 by zero, one or two characters (4 bytes max).  
: 91      0090 1 The routine INIT simply initializes all characters as ignored, the pad  
: 92      0091 1 character as the null character, and no tie-breaking.  
: 93      0092 1  
: 94      0093 1 BASE defines a base collating sequence (via a 256 byte table).  
: 95      0094 1 All 256 single-byte characters are given one-byte collating values,  
: 96      0095 1 taken from the table.  
: 97      0096 1  
: 98      0097 1 NEXT specifies a character that is to get a single-byte collating value  
: 99      0098 1 that collates larger than any other currently defined collating value.  
: 100     0099 1  
: 101     0100 1 OTHERS causes NEXT to be called for all currently ignored single-byte  
: 102     0101 1 characters (similar to COBOL-style definitions).  
: 103     0102 1  
: 104     0103 1 MODIFY defines a character to collate just less than, equal to, or just  
: 105     0104 1 greater than the (0,1,or 2 byte) collating value of a (0,1,or 2 byte)  
: 106     0105 1 character string.  
: 107     0106 1  
: 108     0107 1 FOLD causes all lower case letters to be given the collating values of  
: 109     0108 1 their upper case equivalents. If a double character that contain no  
: 110     0109 1 lower case letters is defined, then lower case and mixed case double  
: 111     0110 1 characters are definded to collate equal to this double character.  
: 112     0111 1 For example,  
: 113     0112 1     This    causes these definitions  
: 114     0113 1     'DS'    "d$"="DS"  
: 115     0114 1     '$D'    "$d"='$d'  
: 116     0115 1     'ds'    none  
: 117     0116 1     'sd'    none  
: 118     0117 1     'xy'    none  
: 119     0118 1     'xy'    none  
: 120     0119 1     'XY'    "xy"="XY","xY"="XY","Xy"="XY"  
: 121     0120 1  
: 122     0121 1 PAD defines the (single byte) pad character.  
: 123     0122 1  
: 124     0123 1  
: 125     0124 1  
: 126     0125 1  
UPPER specifies a simple (i.e., like BASE), secondary collating  
sequence that should be applied if the primary collating sequence  
collates two strings as equal.
```

```
127      0126 1
128      0127 1
129      0128 1
130      0129 1
131      0130 1
132      0131 1
133      0132 1
134      0133 1
135      0134 1
136      0135 1
137      0136 1
138      0137 1
139      0138 1
140      0139 1
141      0140 1
142      0141 1
143      0142 1
144      0143 1
145      0144 1
146      0145 1
147      0146 1
148      0147 1
149      0148 1
150      0149 1
151      0150 1
152      0151 1
153      0152 1
154      0153 1
155      0154 1
156      0155 1
157      0156 1
158      0157 1
159      0158 1
160      0159 1
161      0160 1
162      0161 1
163      0162 1
164      0163 1
165      0164 1
166      0165 1
167      0166 1
168      0167 1
169      0168 1
170      0169 1
171      0170 1
172      0171 1
173      0172 1
174      0173 1
175      0174 1
176      0175 1
177      0176 1
178      0177 1
179      0178 1
180      0179 1
181      0180 1
182      0181 1
183      0182 1
```

TIE_BREAK specifies that, if the primary and secondary collating sequences collate the strings as equal, a final comparison should be done which compares the unsigned binary values of the characters in the strings.

The linkage to the comparison routine is:

(for VAXen):

```
    JSB(
    REGISTER=0,      | Length of String1
    REGISTER=1,      | Address of String1
    REGISTER=2,      | Length of String2
    REGISTER=3,      | Address of String2
    REGISTER=5,      | Address of table
    REGISTER=0):    | Result of the comparison (-1, 0, +1)
    NOPRESERVE(4,5)
    PRESERVE(9,10)
    NOTUSED(6,7,8,11)
```

(for PDP-11s):

TBS

And the condition codes reflect the setting of R0.

IMPLEMENTATION:

During the definition of the collating sequence, collating values are represented by two word values: <x,y>.

<0,0> indicates an ignored character

<x,0> indicates a single-value collating value

<x,y> indicates a double-value collating value

The collating values for single characters are stored in a 256 element array. Double characters and their collating values are stored in a sequential list at the end of the other tables. (The UPPER table is always left in byte form).

The succinct tables generated by RESULT have the form:

RES\$TB	A byte of flags for tie-break and upper
RES\$REVERSE	A byte of flags to reverse sense of tie-break CMPC
RES\$PAD	A byte for the pad character
RES\$PTAB	A 256 byte table, with a value of zero indicating that the STAB table must be consulted.
RES\$UPPER	A 256 byte table.
RES\$STAB	A list of entries for double characters and characters with double collating values. If a character with a value of 0 in PTAB is not found in this table, it is ignored.

The flags within TB are:

TBS\$NOTB	XB'0100'	Don't do tie-breaking (the Z-bit)
TBS\$NOUPPER	XB'0010'	Don't use upper table (the V-bit)

The flags within REVERSE are:

TBS\$REVERSE	XB'0001'	Reverse tie-break CMPC (the C-bit)
--------------	----------	------------------------------------

An entry in STAB is four bytes in length:

<ch0, ch1, cv0, cv1>

The ch0, ch1 together form a single or double character.

The cv0, cv1 together form a single or double collating value.

184 0183 1 | with special forms mentioned above.
185 0184 1 | These entries are ordered in groups with equal ch0 values, in order
186 0185 1 | of increasing ch0 values. The groups are followed by two "trailer"
187 0186 1 | entries:
188 0187 1 | <XX'FF',XX'FF'.....>
189 0188 1 | <XX'00',XX'00'.....>
190 0189 1 | Each group has the form:
191 0190 1 | <x,XX'FF', collating value of x> One of these
192 0191 1 | <x, y , collating value of xy> 0 or more, ordered by y
193 0192 1 |
194 0193 1 | The choice of this representation is succinct and allows for efficient
195 0194 1 | processing. See the support routines SOR\$COLLATE_0, 1 and 2 for more
196 0195 1 | details.
197 0196 1 |
198 0197 1 |--
199 0198 1 LIBRARY 'SYSSLIBRARY:XPORT';
200 0199 1 %IF %BLISS(BLISS32) %THEN
201 0200 1 PSECT
202 0201 1 CODE= SOR\$RO_CODE(PIC,SHARE),
203 0202 1 PLIT= SOR\$RO_CODE(PIC,SHARE);
204 0203 1 XFI
205 0204 1
206 0205 1 LITERAL
207 0206 1 CS_K_REG = %IF %BLISS(BLISS32) %THEN 10 %ELSE 3 %FI;
208 0207 1 MACRO
209 0208 1 LNK_CALL = %IF %BLISS(BLISS32) %THEN CALL %ELSE JSR %FI %;
210 0209 1 LNK_SUBR = %IF %BLISS(BLISS32) %THEN JSB %ELSE JSR %FI %;
211 0210 1 LINKAGE
212 0211 1 CS_LINK_0 = LNK_SUBR: GLOBAL(CS=CS_K_REG),
213 0212 1 CS_CALL_0 = LNK_CALL: GLOBAL(CS=CS_K_REG),
214 0213 1 CS_LINK_1 = LNK_SUBR(REGISTER=1): GLOBAL(CS=CS_K_REG),
215 0214 1 CS_LINK_2 = LNK_SUBR(REGISTER=1, REGISTER=2): GLOBAL(CS=CS_K_REG);
216 0215 1
217 0216 1 FORWARD ROUTINE
218 0217 1 D_LOOKUP: CS_LINK_1. | Look up a double character
219 0218 1 D_NEW: CS_LINK_1. | Create a new secondary table entry
220 0219 1 GIVE_COLL: CS_LINK_2. | Assign a collating value to a character
221 0220 1 DO_BUMP: CS_LINK_1. | Increase collating values
222 0221 1 COLL\$INIT. | Initialize collating sequence
223 0222 1 COLL\$BASE. | Define the base collating sequence
224 0223 1 COLL\$NEXT. | Define the next character
225 0224 1 ! COLL\$OTHERS. | Define undefined single characters
226 0225 1 COLL\$MODIFY. | Make a modification
227 0226 1 COLL\$FOLD. | Fold upper/lower case characters
228 0227 1 COLL\$TIE_BREAK. | Indicate tie-breaking
229 0228 1 COLL\$PAD. | Indicate collating value of the pad character
230 0229 1 COLL\$UPPER. | Upper case comparison
231 0230 1 COLL_VALUE: CS_LINK_2. | Gets the collating value of a character
232 0231 1 COMPRESS: CS_LINK_1. | Compress the range of collating values
233 0232 1 COMPRESS_M: CS_LINK_0. | Compress the tables and set attributes
234 0233 1 COLL\$RESULT; | Build the final tables
235 0234 1
236 0235 1 ! Define the error statuses returned by these routines
237 0236 1
238 0237 1 %IF %BLISS(BLISS32) %THEN
239 0238 1
240 0239 1

```
241      0240 1 EXTERNAL LITERAL
242      0241 1 SORS_COL_ADJ,
243      0242 1 SORS_COL_CMPLX,
244      0243 1 SORS_COL_CHAR,
245      0244 1 SORS_COL_PAD,
246      0245 1 SORS_COL_THRÉE;
247
248      0247 1 BIND
249      0248 1 COLLS_ADJ = SORS_COL_ADJ,          | Invalid ADJ parameter
250      0249 1 COLLS_CMPLX = SORS_COL_CMPLX,    | Collating sequence is too complex
251      0250 1 COLLS_CHAR = SORS_COL_CHAR,       | Invalid character definition
252      0251 1 COLLS_PAD = SORS_COL_PAD,         | Invalid pad character
253      0252 1 COLLS_THREE = SORS_COL_THRÉE;     | Cannot define 3-byte collating values
254
255      0254 1 LITERAL
256      0255 1 TRUE = 1,
257      0256 1 FALSE = 0;
258
259      U 0258 1 XELSE
260      U 0259 1
261      U 0260 1 LIBRARY 'S11V3SRC:SMCOM';
262      U 0261 1
263      U 0262 1 BIND
264      U 0263 1 COLLS_ADJ = SORS_SPCADJ,          | Invalid ADJ parameter
265      U 0264 1 COLLS_CMPLX = SORS_WKAREA,        | Collating sequence is too complex
266      U 0265 1 COLLS_CHAR = SORS_SPCCHR,        | Invalid character definition
267      U 0266 1 COLLS_PAD = SORS_SPCPAD,         | Invalid pad character
268      U 0267 1 COLLS_THREE = SORS_SPCTHR;       | Cannot define 3-byte collating values
269
270      U 0269 1 XFI
271
272
273      0272 1 ! Define the successful status returned by these routines
274
275      0274 1 XIF NOT XDECLARED(SSS_NORMAL) XTHEN LITERAL SSS_NORMAL = 1; XFI
276
277      0276 1 MACRO
278      M 0277 1 IF_ERROR_( X ) = XIF XBLISS( BLISS16 ) XTHEN IF X NEQ SSS_NORMAL
279      0278 1                      XELSE IF NOT X XFI-%;
280
281      M 0280 1 MACRO
282      M 0281 1 CS_SETUP(PARAM) =
283      M 0282 1 XIF XNULL(PARAM)
284      M 0283 1 XTHEN
285      M 0284 1 EXTERNAL REGISTER CS = CS_K_REG: REF CS_BLOCK
286
287      M 0285 1 XELSE
288      M 0286 1 GLOBAL REGISTER CS = CS_K_REG: REF CS_BLOCK;
289      M 0287 1 CS = .PARAM[1]
290
291      0289 1 XIF XDECLARED(%QUOTE ELIF ) XTHEN UNDECLARE %QUOTE ELIF : XFI
292      0290 1 XIF XDECLARED(%QUOTE BASE_) XTHEN UNDECLARE %QUOTE BASE_: XFI
293
294      0292 1 MACRO
295      0293 1 ELIF=           ELSE IF %,
296      0294 1 BASE_=          0, 0, 0, 0 %;
297
298      M 0295 1 MACRO
299      M 0296 1 MOVE_COLL_ALL_(X,Y) =
```

```
298      M 0297 1      BEGIN
299      M 0298 1      %IF %FIELDEXPAND(COLL_ALL,2) NEQ 0
300      M 0299 1      %THEN
301      M 0300 1          BBLOCK[X,COLL_ALL] = .BBLOCK[Y,COLL_ALL];
302      M 0301 1      %ELSE
303      M 0302 1          BBLOCK[X,COLL_C0] = .BBLOCK[Y,COLL_C0];
304      M 0303 1          BBLOCK[X,COLL_C1] = .BBLOCK[Y,COLL_C1];
305      M 0304 1      %FI
306      M 0305 1      END %;
307      M 0306 1      !MACRO
308      M 0307 1          MOVE32_(X,Y) =
309      M 0308 1          %IF %BLISS(BLISS32)
310      M 0309 1          %THEN X = .Y
311      M 0310 1          %ELSE ((X) = .(Y); (X+2)=.(Y+2)) %FI %
312      M 0311 1      LITERAL
313      M 0312 1          K_CHARS = 256;           ! Number of 1-byte characters
314      M 0313 1
315      M 0314 1      !MACRO
316      M 0315 1          XBYTE =    %EXPAND $BITS(8) %
317      M 0316 1          XWORD =    %EXPAND $BITS(16) %
318      M 0317 1          XLONG =    %EXPAND $BITS(32) %
319      M 0318 1          XDESC =    $SUB_BLOCK(2) %
320      M 0319 1          XADDR =    $ADDRESS %
321      M 0320 1          $SHOW(FIELDS)
322      M 0321 1
323      M 0322 1      STRUCTURE
324      M 0323 1          BBLOCK[0,P,S,E;BS=0] = [BS](BBLOCK+0)<P,S,E>;
```

```
: 326      0324 1      C H A R - B L O C K
: 327      0325 1
: 328      0326 1      A char is an elementary data structure representing a single or double
: 329      0327 1      character.
: 330      0328 1
: 331      0329 1      $UNIT_FIELD
: 332      0330 1      CHAR_FIELDS =
: 333      0331 1      SET
: 334      L 0332 1      CHAR_LEN=      [XWORD],
: 335      L 0333 1      CHAR_C0=      [XBYTE]      (+%X'0')
: 336      L 0334 1      CHAR_C1=      [2,0,8,0]      (+%X'2')
: 337      L 0335 1      CHAR_C2=      [XBYTE]
: 338      L 0336 1      CHAR_C01=     [3,0,8,0]      (+%X'3')
: 339      L 0337 1      $OVERLAY(CHAR_C0)
: 340      L 0338 1      CHAR_ALL=      [XWORD]
: 341      L 0339 1      CHAR_C01=     [2,0,16,0]      (+%X'2')
: 342      L 0340 1      CHAR_ALL=      [XLONG]
: 343      L 0341 1      CHAR_ALL=     [0,0,32,0]      (+%X'0')
: 344      TES;
: 345      LITERAL CHAR_K_SIZE=      $FIELD_SET_UNITS;      ! Size in bytes
: 346      MACRO  CHAR_BLOCK=      BBLOCK[CHAR_K_SIZE] FIELD(CHAR_FIELDS) %;
```

```
345      0342 1          C O L L _ B L O C K
346      0343 1
347      0344 1          A coll is an elementary data structure representing a single, double or
348      0345 1          ignored collating value.
349      0346 1          <0,0>    ignored
350      0347 1          <x,0>    single collating value (x ne 0)
351      0348 1          <x,y>    double collating value (x,y ne 0)
352      0349 1
353      0350 1 $UNIT_FIELD
354      0351 1     COLL_FIELDS =
355      0352 1     SET
356      L 0353 1     COLL_C0=      [XWORD]           [+XX'0']
357      L 0354 1     COLL_C1=      [XWORD]           [+XX'2']
358      0355 1 $OVERLAY(0,0,0,0)
359      L 0356 1     COLL_ALL=     [XLONG]           [0,0,32,0] (+XX'0')
360      0357 1     TES:
361      0358 1 LITERAL COLL_K_SIZE= $FIELD_SET_UNITS; ! Size in bytes
362      0359 1 MACRO   COLL_BLOCK= BBLOCK[COLL_K_SIZE] FIELD(COLL_FIELDS) %;
```

364 0360 1 | C S - B L O C K
365 0361 1 |
366 0362 1 | This data structure holds pertinent information between calls.
367 0363 1 |
368 0364 1 | \$UNIT_FIELD
369 0365 1 | CS_FIELDS =
370 0366 1 | SET
371 L 0367 1 | CS_SIZE= [XWORD], ! Size of this block
XPRINT: 372 L 0368 1 | CS_CURR_SIZE= [XWORD], ! Current size of this block
373 L 0369 1 | CS_COLL_MAX= [XWORD], ! Largest collating value
374 L 0370 1 | CS_DCHAR= [XWORD], ! Number of double characters
375 L 0371 1 | CS_TB= [XBYTE], ! Tie-break / Upper bits
376 L 0372 1 | CS_PAD= [XBYTE], ! Pad character
377 L 0373 1 | CS_REVERSE= [XBYTE], ! Reverse sense of tie-break CMPC
378 L 0374 1 | CS_MODS= [\$BIT], ! Modifications were made
379 L 0375 1 | CS_IGN= [\$BIT], ! There are ignored characters
380 L 0376 1 | CS_DCOLL= [\$BIT], ! There are double collating values
381 0377 1 | \$ALIGN(WORD)
382 0378 1 | CS_PSTATIC= [\$ADDRESS], ! Address of static base table
383 0379 1 | CS_USTATIC= [\$ADDRESS], ! Address of static upper table
384 L 0380 1 | CS_UPPER= [\$BYTES(K_CHARS)], ! Secondary table
XPRINT: 385 L 0381 1 | CS_PTAB= [12,0,0,0] (+XX'C') ! Table of single chars
XPRINT: 386 L 0382 1 | CS_STAB= [268,0,0,0] (+XX'10C') ! Table of double chars
387 0383 1 | TES:
388 0384 1 | LITERAL CS_K_SIZE= \$FIELD SET UNITS: ! Size in bytes
389 0385 1 | MACRO CS_BLOCK= BBLOCK[CS_R_SIZE] FIELD(CS_FIELDS,RES_FIELDS) %;
390 M 0386 1 | MACRO CS_PTAB_(X)= COLL_K_SIZE+(X)+%FIELDEXPAND(CS_PTAB,0),0,
391 M 0387 1 | %IF COLL_K_SIZE+%BPUNIT LEQ %BPVAL
392 0388 1 | %THEN COLL_K_SIZE+%BPUNIT %ELSE 0 %FI,0 %;

```
394 0389 1 | R E S - B L O C K
395 0390 1 |
396 0391 1 | This data structure holds the compressed form of the tables.
397 0392 1 | For Bliss-11, it is defined in a library so that the structure can be known
398 0393 1 | to the comparison routines, which are in a different overlay.
399 0394 1 |
400 U 0395 1 %IF NOT %BLISS(BLISS32) %THEN
401 U 0396 1 |
402 U 0397 1 LIBRARY 'S11V3SRC:SORCOLUTI';
403 U 0398 1 |
404 U 0399 1 %ELSE
405 U 0400 1 |
406 U 0401 1 $UNIT FIELD
407 U 0402 1     RES_FIELDS =
408 U 0403 1       SET
409 L 0404 1         RES_RTN=     [$ADDRESS],     [0,0,32,0]     (+%X'0')
410 L 0405 1         RES_RTN_A=    [$ADDRESS],     [4,0,32,0]     (+%X'4')
411 L 0406 1         RES_TB=     [$BYTE],     [8,0,8,0]     (+%X'8')
412 L 0407 1         RES_PAD=    [$BYTE],     [9,0,8,0]     (+%X'9')
413 L 0408 1         RES_REVERSE=    [$BYTE],     [10,0,8,0]     (+%X'A')
414      0409 1         $ALIGN(WORD)
415 L 0410 1         RES_PTAB=    [$BYTES(K_CHARS)],     [12,0,0,0]     (+%X'C')
416 L 0411 1         RES_UPPER=    [$BYTES(K_CHARS)],     [268,0,0,0]     (+%X'10C')
417 L 0412 1         RES_STAB=    [$BYTES(0)]     [524,0,0,0]     (+%X'20C')
418      0413 1         TES;
419      0414 1         LITERAL RES_K_SIZE=     $FIELD_SET_UNITS;     : Size in bytes
420      0415 1 |
421      0416 1 %FI
422      0417 1 |
423      0418 1 %IF RES_K_SIZE GTR CS_K_SIZE %THEN %ERROR('Something terrible happened') %FI
424      0419 1 |
425      0420 1 | These values must be known to the macro routine
426      0421 1 |
427      0422 1 GLOBAL LITERAL
428      0423 1         RESSRTN=     %FIELDEXPAND(RES_RTN,0),
429      0424 1         RESSTB=     %FIELDEXPAND(RES_TB,0),
430      0425 1         RESSREVERSE=    %FIELDEXPAND(RES_REVERSE,0),
431      0426 1         RESSPAD=    %FIELDEXPAND(RES_PAD,0),
432      0427 1         RESSPTAB=    %FIELDEXPAND(RES_PTAB,0),
433      0428 1         RESSUPPER=    %FIELDEXPAND(RES_UPPER,0),
434      0429 1         RESSSTAB=    %FIELDEXPAND(RES_STAB,0),
435      0430 1         TBSNOTB =    %B'0100';     ! Don't do tie-breaking (the Z-bit)
436      0431 1         TBSNOUPPER =    %B'0010';     ! Don't use upper table (the V-bit)
437      0432 1         TBSREVERSE =    %B'0001';     ! Reverse tie-break CMPC (the C-bit)
438      0433 1 |
```

```
: 440      0434 1 |          S T - B L O C K
: 441      0435 1 |
: 442      0436 1 |          A secondary table entry consists of:
: 443      0437 1 |              An indication whether the input character is one or two bytes.
: 444      0438 1 |              The one or two byte input character.
: 445      0439 1 |              The collating value.
: 446      0440 1 |              The offset to the next secondary table entry.
: 447      0441 1 |
: 448      0442 1 $UNIT_FIELD
: 449      0443 1     ST_FIELDS =
: 450      0444 1     SET
: 451      L 0445 1     ST_CHAR=      [XWORD],
: %PRINT:   L 0446 1     ST_COLL=      [0,0,16,0] (+XX'0')
:           L 0446 1             [$_BYTES(COLL_K_SIZE)],
: %PRINT:   0447 1 $OVERLAY(ST_CHAR)
:           L 0448 1     ST_CHAR_0=    [XBYTE],
: %PRINT:   L 0449 1     ST_CHAR_1=    [0,0,8,0] (+XX'0')
:           L 0449 1             [XBYTE]
: %PRINT:   0450 1     ST_CHAR_1=    [1,0,8,0] (+XX'1')
: 456      0450 1 $CONTINUE
: 457      0451 1     TES:
: 458      0452 1 LITERAL ST_K_SIZE=    $FIELD SET_UNITS;
:           0453 1 MACRO  ST_BLOCK=      BBLOCK[ST_R_SIZE] FIELD(ST_FIELDS) %;
```

: 461 0454 1 | When we are inserting another collating value, but have no available
: 462 0455 1 | single-byte collating values, we can:
: 463 0456 1 |
: 464 0457 1 | Find two adjacent one-byte collating values (x and x+1) that:
: 465 0458 1 | are not used as the first byte of any two-byte collating values,
: 466 0459 1 | and are not used as the second byte of any two-byte collating values
: 467 0460 1 | for which the first byte is used as a one-byte collating value.
: 468 0461 1 | Change the characters that collate to x and x+1 to collate to the two-byte
: 469 0462 1 | collating values <x,0> and <x,1>, respectively.
: 470 0463 1 | This frees the single-byte collating value x+1.
: 471 0464 1 |
: 472 0465 1 | Or (preferably) we can:
: 473 0466 1 |
: 474 0467 1 | Find a collating value (x) that:
: 475 0468 1 | is used only as the first byte of two-byte collating values
: 476 0469 1 | for which not all 256 different second byte values are used,
: 477 0470 1 | it has an adjacent value y (either x-1 or x+1) such that:
: 478 0471 1 | it is not used as the first byte of any two-byte collating values,
: 479 0472 1 | and is not used as the second byte of any two-byte collating values
: 480 0473 1 | for which the first byte is used as a one-byte collating value.
: 481 0474 1 | Add another second-byte collating value (z) to the two-byte collating
: 482 0475 1 | values that have x as their first-byte collating value, such that z is
: 483 0476 1 | less than (y=x-1), or greater than (y=x+1) all the other second-byte
: 484 0477 1 | collating values.
: 485 0478 1 | Change the characters that collate to y to collate to <x,z>.

```
: 487      0479 1 GLOBAL ROUTINE COLL$INIT(          :  
: 488      0480 1     COLL_SEQ:      REF VECTOR[2]           ! Collating sequence  
: 489      0481 1     ) =  
: 490      0482 1     ++  
: 491      0483 1  
: 492      0484 1     FUNCTIONAL DESCRIPTION:  
: 493      0485 1  
: 494      0486 1     Initialize a collating sequence description.  
: 495      0487 1     It is initialized to all ignored characters.  
: 496      0488 1  
: 497      0489 1     FORMAL PARAMETERS:  
: 498      0490 1  
: 499      0491 1     COLL_SEQ      a two-longword array specifying the length/address  
: 500      0492 1     of storage to use for the collating sequence.  
: 501      0493 1  
: 502      0494 1     IMPLICIT INPUTS:  
: 503      0495 1  
: 504      0496 1     NONE  
: 505      0497 1  
: 506      0498 1     IMPLICIT OUTPUTS:  
: 507      0499 1  
: 508      0500 1     The memory specified by COLL_SEQ is initialized.  
: 509      0501 1  
: 510      0502 1     ROUTINE VALUE:  
: 511      0503 1  
: 512      0504 1     Status code  
: 513      0505 1  
: 514      0506 1     SIDE EFFECTS:  
: 515      0507 1  
: 516      0508 1     NONE  
: 517      0509 1  
: 518      0510 1     --  
: 519      0511 2     BEGIN  
: 520      0512 2  
: 521      0513 2     CS_SETUP(COLL_SEQ);  
: 522      0514 2  
: 523      0515 2     IF .COLL_SEQ[0] LSSU CS_K_SIZE THEN RETURN COLL$_CMPLX;  
: 524      0516 2     CH$FILL(0, CS_K_SIZE, CS[BASE]);  
: 525      0517 2     CS[CS_SIZE] = MINU(.COLL_SEQ[0], 1^%FIELDEXPAND(CS_SIZE,2)-1);  
: 526      0518 2     CS[CS_CURR_SIZE] = CS_K_SIZE;  
: 527      0519 2     CS[CS_TB] = TBSNOTB OR TBSNOUPPER;  
: 528      0520 2  
: 529      0521 2     RETURN SSS_NORMAL;  
: 530      0522 1     END;
```

.TITLE COLL\$UTILITIES
.IDENT \V04-000\

RES\$RTN==	0
RES\$TB==	8
RES\$REVERSE==	10
RES\$PAD==	9
RES\$PTAB==	12
RES\$UPPER==	268
RES\$STAB==	524
TBSNOTB==	4

```

TBSNOUPPER==          2
TBSREVERSE==          1

        .EXTRN  SORS_COL_ADJ, SORS_COL_CMPLX
        .EXTRN  SORS_COL_CHAR, SORS_COL_PAD
        .EXTRN  SORS_COL_THREE

        .PSECT  SOR$RO_CODE,NOWRT,  SHR,  PIC,2

        .ENTRY  COL$INIT, Save R2,R3,R4,R5,R6,R10      : 0479
        MOVL    COLL SEQ, R6                           : 0513
        MOVL    4(R6), CS
        CMPL    (R6), #1292                           : 0515
        BGEQU   1$
        MOVL    #COLL$_CMPLX, R0
        RET
1$:     MOVCS   #0, (SP), #0, #1292, (CS)           : 0516
        MOVL    (R6), R0
        CMPL    R0, #65535                            : 0517
        BLEQU   2$
        MOVZWL  #65535, R0
        MOVW    R0, (CS)
        MOVW    #1292, 2(CS)
        MOVB    #6, 8(CS)
        MOVL    #1, R0
        RET
2$:     MOVL    R0, (CS)
        MOVB    #1, 8(CS)
        MOVL    R0, (CS)
        RET

```

; Routine Size: 69 bytes, Routine Base: SOR\$RO_CODE + 0000

```
: 532      0523 1 GLOBAL ROUTINE COLL$BASE(
: 533      0524 1     COLL_SEQ:    REF VECTOR[2],           ! Collating sequence
: 534      0525 1     BASE_SEQ:    REF VECTOR[K_CHARS,BYTE]   ! Base sequence
: 535      0526 1     STATIC
: 536      0527 1     S =
: 537      0528 1     ++
: 538      0529 1
: 539      0530 1     FUNCTIONAL DESCRIPTION:
: 540      0531 1
: 541      0532 1     Specify the base collating sequence.
: 542      0533 1
: 543      0534 1     FORMAL PARAMETERS:
: 544      0535 1
: 545      0536 1     COLL_SEQ      a two-longword array specifying the length/address
: 546      0537 1             of storage to use for the collating sequence.
: 547      0538 1
: 548      0539 1     BASE_SEQ      a 256-byte array giving the (single byte) collating
: 549      0540 1             value for each character.
: 550      0541 1
: 551      0542 1     IMPLICIT INPUTS:
: 552      0543 1
: 553      0544 1     INIT must have already been called.
: 554      0545 1
: 555      0546 1     IMPLICIT OUTPUTS:
: 556      0547 1
: 557      0548 1     NONE
: 558      0549 1
: 559      0550 1     ROUTINE VALUE:
: 560      0551 1
: 561      0552 1     Status code
: 562      0553 1
: 563      0554 1     SIDE EFFECTS:
: 564      0555 1
: 565      0556 1     NONE
: 566      0557 1
: 567      0558 1     --
: 568      0559 2     BEGIN
: 569      0560 2     LOCAL
: 570      0561 2     BS:    REF VECTOR[K_CHARS,BYTE];
: 571      0562 2     BUILTIN
: 572      0563 2     NULLPARAMETER;
: 573      0564 2
: 574      0565 2     CS_SETUP(COLL_SEQ);
: 575      0566 2
: 576      0567 2     BS = BASE_SEQ[0];
: 577      0568 2     DECR I FROM K_CHARS-1 TO 0 DO (CS[CS_PTAB_(.I)]) = .BS[.I] + 1;
: 578      0569 2     CS[CS_COLL_MAX] = K_CHARS;
: 579      0570 2     ! IF NOT NUL[PARAMETER(3) THEN CS[CS_PSTATIC] = BASE_SEQ[0];
: 580      0571 2
: 581      0572 2     RETURN SSS_NORMAL;
: 582      0573 1     END;
```

50	04	AC	7D	00002	MOVQ	COLL SEQ, R0	: 0565
5A	04	A0	D0	00006	MOVL	4(R0), CS	
50	FF	8F	9A	0000A	MOVZBL	#255, I	: 0568
010C CA40	6041	9A	0000E	1\$:	MOVZBL	(I)[BS], 268(CS)[I]	
F1	010C CA40	D6	00015		INCL	268(CS)[I]	
04 AA	0100	8F	B0	0001D	S0BGEQ	I, 1\$: 0569
50	01	00	00023		MOVW	#256, 4(CS)	: 0572
		04	00026		MOVL	#1, R0	
					RET		: 0573

: Routine Size: 39 bytes, Routine Base: SOR\$R0_CODE + 0045

```
584      0574 1 GLOBAL ROUTINE COLL$UPPER(
585      0575 1   COLL_SEQ:      REF VECTOR[2],
586      0576 1   UPPER_SEQ:    REF VECTOR[K_CHARS,BYTE]      ! Collating sequence
587      0577 1   !           STATIC                      ! Secondary sequence
588      0578 1   { = ++
589
590      0580 1
591      0581 1   FUNCTIONAL DESCRIPTION:
592
593      0583 1   Specify the secondary collating sequence.
594      0584 1   If two strings compare equal using the sequence specified with BASE,
595      0585 1   SEQUENCE, MODIFY and IGNORE, the collating sequence specified by this
596      0586 1   routine is then used.
597
598      0587 1   FORMAL PARAMETERS:
599
600      0589 1   COLL_SEQ      a two-longword array specifying the length/address
601      0590 1   of storage to use for the collating sequence.
602
603      0592 1   UPPER_SEQ     a 256-byte array giving the (single byte) collating
604      0593 1   value for each character.
605
606      0595 1   IMPLICIT INPUTS:
607
608      0597 1   INIT must have already been called.
609
610      0599 1   IMPLICIT OUTPUTS:
611
612      0601 1   NONE
613
614      0603 1   ROUTINE VALUE:
615
616      0605 1   Status code
617
618      0607 1   SIDE EFFECTS:
619
620      0609 1   NONE
621
622      0611 1   !-- BEGIN
623      0613 2 LOCAL
624      0614 2   BS:      REF VECTOR[K_CHARS,BYTE].
625
626      0615 2   X:
627
628      0617 2   BUILTIN
629      0618 2   NULLPARAMETER;
630
631      0619 2   CS_SETUP(COLL_SEQ);
632
633      0620 2   X = UPPER_SEQ[0];
634      0621 2   IF .X NEQ 0 THEN X = K_CHARS;
635      0622 2   CH$COPY(.X, UPPER_SEQ[0], 0, K_CHARS, CS[CS_UPPER]);
636
637      0623 2   ! IF NOT NULLPARAMETER(3) THEN CS[CS_USTATIC] = UPPER_SEQ[0];
638      0624 2   CS[CS_TB] = .CS[CS_TB] AND NOT TB$NOUPPER;
639
640      0625 2   RETURN SSS_NORMAL;
641
642      0626 1   END;
```

0100	8F		00	08	50	0100	043C	00000	.ENTRY	COLL\$UPPER, Save R2,R3,R4,R5,R10	: 0574
					50	04	AC	00 0002	MOVL	COLL SEQ, RC	: 0620
					5A	04	A0	00 0006	MOVL	4(R0), CS	: 0621
					50	08	AC	00 000A	MOVL	UPPER_SEQ, X	: 0622
						05	13	0000E	BEQL	1\$: 0623
						8F	3C	00010	MOVZWL	#256, X	: 0624
						50	2C	00015	MOVC5	X, @UPPER_SEQ, #0, #256, 12(CS)	: 0625
						0C	AA	0001D			
						08	AA	02 8A 0001F	BICB2	#2, 8(CS)	: 0627
							50	01 D0 00023	MOVL	#1, R0	: 0629
								04 00026	RET		: 0630

; Routine Size: 39 bytes, Routine Base: SOR\$RO_CODE + 006C

```

642      0631 1 GLOBAL ROUTINE COLL$NEXT(
643          0632 1     COLL_SEQ:      REF VECTOR[2],
644          0633 1     CHART:        REF CHAR_BLOCK
645          0634 1     ) =
646          0635 1     ++
647          0636 1
648          0637 1     FUNCTIONAL DESCRIPTION:
649          0638 1
650          0639 1     Define a character to collate greater than any currently defined
651          0640 1     character.
652          0641 1
653          0642 1     FORMAL PARAMETERS:
654          0643 1
655          0644 1     COLL_SEQ      a two-longword array specifying the length/address
656          0645 1     of storage to use for the collating sequence.
657          0646 1
658          0647 1     CHAR1        a character.
659          0648 1
660          0649 1     IMPLICIT INPUTS:
661          0650 1
662          0651 1     INIT must have already been called.
663          0652 1
664          0653 1     IMPLICIT OUTPUTS:
665          0654 1
666          0655 1     NONE
667          0656 1
668          0657 1     ROUTINE VALUE:
669          0658 1
670          0659 1     Status code
671          0660 1
672          0661 1     SIDE EFFECTS:
673          0662 1
674          0663 1     NONE
675          0664 1
676          0665 1     --
677          0666 2     BEGIN
678          0667 2     LOCAL
679          0668 2     COLL:    COLL_BLOCK;
680          0669 2
681          0670 2     CS_SETUP(COLL_SEQ);
682          0671 2
683          0672 2     CS[CS_MODS] = TRUE;
684          0673 2
685          0674 2     CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] + 1;
686          0675 2     COLL[COLL_C0] = .CS[CS_COLL_MAX];
687          0676 2     COLL[COLL_C1] = 0;
688          0677 2     RETURN GIVE_COLL( CHAR1[CHAR_ALL], COLL[COLL_ALL] );
689          0678 1     END;

```

	OFFC 00000	.ENTRY	COLL\$NEXT, Save R2,R3,R4,R5,R6,R7,R8,R9,-	: 0631
50 5A	04 04 AC A0 7D D0 00002	MOVQ	COLL SEQ, R0	: 0670
	00006	MOVL	4(R0), CS	

COLL\$UTILITIES
V04-000

I 12
16-Sep-1984 01:06:02 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:40 [SORT32.SRC]SORCOLUTI.B32;1

Page 21
(12)

0B AA	01 88 0000A	BISB2 #1, 11(CS)	: 0672
7E	04 AA B6 0000E	INCW 4({S})	: 0674
52	04 AA 3C 00011	MOVZWL 4(CS), COLL	: 0675
	6E 9E 00015	MOVAB COLL, R2	: 0677
	0000V 30 00018	BSBW GIVE_COLL	
	04 0001B	RET	: 0678

: Routine Size: 28 bytes, Routine Base: SOR\$RO_CODE + 0093

```
691 C 0679 1 %(  
692 C 0680 1 GLOBAL ROUTINE COLL$OTHERS(  
693 C 0681 1     COLL_SEQ:      REF VECTOR[2]           ' Collating sequence  
694 C 0682 1     ) =  
695 C 0683 1 ++  
696 C 0684 1  
697 C 0685 1 FUNCTIONAL DESCRIPTION:  
698 C 0686 1  
699 C 0687 1 Define all currently ignored (undefined) characters to collate larger  
700 C 0688 1 than all the non-ignored (defined) characters, in order of the  
701 C 0689 1 character codes.  
702 C 0690 1  
703 C 0691 1 FORMAL PARAMETERS:  
704 C 0692 1  
705 C 0693 1     COLL_SEQ      a two-longword array specifying the length/address  
706 C 0694 1             of storage to use for the collating sequence.  
707 C 0695 1  
708 C 0696 1 IMPLICIT INPUTS:  
709 C 0697 1  
710 C 0698 1     INIT must have already been called.  
711 C 0699 1  
712 C 0700 1 IMPLICIT OUTPUTS:  
713 C 0701 1  
714 C 0702 1  
715 C 0703 1  
716 C 0704 1  
717 C 0705 1  
718 C 0706 1  
719 C 0707 1  
720 C 0708 1  
721 C 0709 1  
722 C 0710 1  
723 C 0711 1  
724 C 0712 1 --  
725 C 0713 1 BEGIN  
726 C 0714 1 LOCAL  
727 C 0715 1     CHAR:  CHAR_BLOCK,  
728 C 0716 1     P:    REF COLL_BLOCK,  
729 C 0717 1     S:  
730 C 0718 1  
731 C 0719 1  
732 C 0720 1  
733 C 0721 1  
734 C 0722 1  
735 C 0723 1  
736 C 0724 1  
737 C 0725 1  
738 C 0726 1  
739 C 0727 1  
740 C 0728 1  
741 C 0729 1  
742 C 0730 1  
743 C 0731 1  
744 C 0732 1  
745 C 0733 1  
746 C 0734 1  
747 C 0735 1  
CS_SETUP(COLL_SEQ);  
CSE[CS_MODS] = TRUE;  
CHAR[CHAR_LEN] = 1;  
P = CS[CS_PTAB];  
INCR I FROM 0 TO K_CHARS-1 DO  
    BEGIN  
        IF  
            XIF XFIELDEXPAND(COLL_ALL,2) NEQ 0  
            XTHEN .P[COLL_ALL] EQ[ 0  
            XELSE .P[COLL_C0] EQ[ 0 AND .P[COLL_C1] EQ[ 0  
            XFI  
        THEN  
            BEGIN  
                CHAR[CHAR_C0] = .I;
```

K 12
16-Sep-1984 01:06:02
14-Sep-1984 13:10:40VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SORCOLUTI.B32;1Page 23
(13)

```
748      C 0736 1      S = COLL$NEXT(COLL_SEQ[0], CHAR[BASE_]);  
.: 749      C 0737 1      IF ERROR_( .S ) THEN RETURN .S;  
.: 750      C 0738 1      END;  
.: 751      C 0739 1      P = .P + COLL_K_SIZE;  
.: 752      C 0740 1      END;  
.: 753      C 0741 1      RETURN SSS_NORMAL;  
.: 754      C 0742 1      END;  
.: 755      C 0743 1      END;  
.: 756      C 0744 1  )%
```

```
758      0745 1 MACRO
759      M 0746 1   FOR_ALL_COLLSS(X) =
760      M 0747 1     BEGIN
761      M 0748 1       LOCAL X: REF COLL_BLOCK;
762      M 0749 1       LOCAL STEP;
763      M 0750 1       X = CS[CS_PTAB];
764      M 0751 1       STEP = COLL_K_SIZE;
765      M 0752 1       DECR FIRST FROM 1 TO 0 DO
766      M 0753 1         BEGIN
767      M 0754 1           DECR I FROM (IF .FIRST THEN K_CHARS ELSE .CS[CS_DCHAR])-1 TO 0 DO
768      M 0755 1             BEGIN
769      M 0756 1               X,
770      M 0757 1             END_ALL_COLLSS(X) =
771      M 0758 1               X = .X + .STEP;
772      M 0759 1             END;
773      M 0760 1               STEP = ST_K_SIZE;
774      M 0761 1               X = .X + %FIELDEXPAND(ST_COLL,0)
775      M 0762 1                 - K_CHARS * COLL_K_SIZE
776      M 0763 1                 - %FIELDEXPAND(CS_PTAB,0)
777      M 0764 1                 + %FIELDEXPAND(CS_STAB,0);
778      M 0765 1             END;
779      M 0766 1           END %,
780      M 0767 1           FOR_ALL_DCHARS(X) =
781      M 0768 1             BEGIN
782      M 0769 1               LOCAL X: REF ST_BLOCK;
783      M 0770 1               X = CS[CS_STAB];
784      M 0771 1               DECR I FROM .CS[CS_DCHAR]-1 TO 0 DO
785      M 0772 1                 BEGIN
786      M 0773 1                   X,
787      M 0774 1                 END_ALL_DCHARS(X) =
788      M 0775 1                   X = .X + ST_K_SIZE;
789      M 0776 1                 END;
790      M 0777 1             END %,
791      M 0778 1             FOR_ALL_SCHARS(X) =
792      M 0779 1               BEGIN
793      M 0780 1                 LOCAL X: REF COLL_BLOCK;
794      M 0781 1                 X = CS[CS_PTAB];
795      M 0782 1                 DECR I FROM K_CHARS-1 TO 0 DO
796      M 0783 1                   BEGIN
797      M 0784 1                       X,
798      M 0785 1                     END_ALL_SCHARS(X) =
799      M 0786 1                     X = .X + COLL_K_SIZE;
800      M 0787 1                   END;
801      M 0788 1             END %;
```

```
803      0789 1 GLOBAL ROUTINE COLLSMODIFY(
804          0790 1     COLL_SEQ:           REF VECTOR[2],
805          0791 1     CHAR1:            REF CHAR_BLOCK,
806          0792 1     CHAR2:            REF CHAR_BLOCK,
807          0793 1     ADJ:
808          0794 1     ) =
809          0795 1 ++
810          0796 1
811          0797 1     FUNCTIONAL DESCRIPTION:
812          0798 1
813          0799 1     Modify the collating sequence.
814          0800 1     Based on the value of ADJ, define CHAR1 to collate just less than (-1),
815          0801 1     equal to (0), or just greater than (+1) CHAR2.
816          0802 1
817          0803 1     FORMAL PARAMETERS:
818          0804 1
819          0805 1     COLL_SEQ      a two-longword array specifying the length/address
820          0806 1           of storage to use for the collating sequence.
821          0807 1
822          0808 1     CHAR1        the character being defined.
823          0809 1
824          0810 1     CHAR2        the character used to define CHAR1.
825          0811 1
826          0812 1     ADJ          adjustment; either -1, 0 or +1.
827          0813 1
828          0814 1     IMPLICIT INPUTS:
829          0815 1
830          0816 1     INIT must have already been called.
831          0817 1
832          0818 1     IMPLICIT OUTPUTS:
833          0819 1
834          0820 1     NONE
835          0821 1
836          0822 1     ROUTINE VALUE:
837          0823 1
838          0824 1     Status code
839          0825 1
840          0826 1     SIDE EFFECTS:
841          0827 1
842          0828 1
843          0829 1
844          0830 1     !!--
845          0831 2     BEGIN LOCAL
846          0832 2
847          0833 2     COLL:    COLL_BLOCK,
848          0834 2     LADJ,
849          0835 2     S;
850          0836 2
851          0837 2     CS_SETUP(COLL_SEQ);
852          0838 2
853          0839 2     CSE[CS_MODS] = TRUE;
854          0840 2
855          0841 2     !+
856          0842 2     Define CHAR1 to collate:
857          0843 2     (ADJ = -1) less than, (ADJ = 0) equal to, or (ADJ = +1) greater than
858          0844 2     the character CHAR2.
859          0845 2     !-
```

```
860      0846 2
861      0847 2  | Check that ADJ = +1, 0, or -1
862      0848 2
863      0849 2  LADJ = .ADJ;
864      0850 2  SELECTONE .LADJ OF SET [-1,0,+1]:0; [OTHERWISE]:RETURN COLLS_ADJ; TES;
865      0851 2
866      0852 2
867      0853 2  | Set COLL to the current collating value of CHAR2
868      0854 2
869      0855 2  S = COLL_VALUE(CHAR2[CHAR_ALL], COLLE[COLL_ALL]);
870      0856 2  IF_ERROR_( .S ) THEN RETURN .S;
871      0857 2
872      0858 2
873      0859 2  | If COLL indicates an ignored character
874      0860 2  Then
875      0861 2    Check that ADJ >= 0
876      0862 2    If ADJ > 0 then set COLL to the lowest character, and ADJ to -1
877      0863 2
878      0864 2  IF .COLL[COLL_C0] EQ 0
879      0865 2  THEN
880      0866 3    BEGIN
881      0867 3    IF .LADJ LSS 0 THEN RETURN COLLS_ADJ;
882      0868 3    IF .LADJ GTR 0
883      0869 3    THEN
884      0870 4    BEGIN
885      0871 4    COLL[COLL_C0] = 1;           | The smallest collating value
886      0872 4    COLL[COLL_C1] = 0;           | No second character
887      0873 4    LADJ = -1;                 | Create something even smaller
888      0874 3    END;
889      0875 2    END;

890      0876 2
891      0877 2
892      0878 2  | Give CHAR1 the collating value COLL
893      0879 2
894      0880 2  S = GIVE_COLL( CHAR1[CHAR_ALL], COLL[COLL_ALL] );
895      0881 2  IF_ERROR_( .S ) THEN RETURN .S;
896      0882 2
897      0883 2
898      0884 2  | If ADJ = 0 then we are done
899      0885 2
900      0886 2  IF .LADJ EQ 0 THEN RETURN SSS_NORMAL;
901      0887 2
902      0888 2
903      0889 2  | Set COLL to the current collating value of CHAR1
904      0890 2
905      0891 2  S = COLL_VALUE(CHAR1[CHAR_ALL], COLLE[COLL_ALL]);
906      0892 2  IF_ERROR_( .S ) THEN RETURN .S;
907      0893 2
908      0894 2
909      0895 2  | Bump the collating values of everything greater than or equal to the
910      0896 2  new collating value we want to give CHAR1.
911      0897 2
912      0898 2  IF (S = .COLL[COLL_C1]) EQ 0 THEN S = .COLL[COLL_C0];
913      0899 2  IF .LADJ GTR 0 THEN S = .S + 1;
914      0900 2  S = DO_BUMP(.S);
915      0901 2  IF ERROR_( .S ) THEN RETURN .S;
916      0902 2  CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] + 1;
```

```

917    0903 2
918    0904 2
919    0905 2
920    0906 2
921    0907 2
922    0908 2
923    0909 2
924    0910 2
925    0911 2
926    0912 2
927    0913 2
928    0914 2
929    0915 2
930    0916 2
931    0917 2
932    0918 1

      ! Set COLL to the current collating value of CHAR1
      S = COLL_VALUE(CHAR1[CHAR_ALL], COLL[COLL_ALL]);
      IF_ERROR_( .S ) THEN RETURN .S;

      ! Adjust the collating value COLL, and assign it to CHAR1
      IF .COLL[COLL_C1] NEQ 0
      THEN COLL[COLL_C1] = .COLL[COLL_C1] + .LADJ
      ELSE COLL[COLL_C0] = .COLL[COLL_C0] + .LADJ;
      RETURN GIVE_COLL( CHAR1[CHAR_ALC], COLL[COLL_ALL] );
END;

```

			OFFC 00000	.ENTRY	COLL\$MODIFY. Save R2,R3,R4,R5,R6,R7,R8,R9,- : 0789
	54	0000V	CF 9E 00002	MOVAB	COLL VALUE, R4
	5E		04 C2 00007	SUBL2	#4, SP
	50	04	AC D0 0000A	MOVL	COLL SEQ, R0
	5A	04	A0 D0 0000E	MOVL	4(R0), CS
0B	AA		01 88 00012	BISB2	#1, 1f(CS)
	53	10	AC D0 00016	MOVL	ADJ, LADJ
FFFFFFF	8F		53 D1 0001A	CMPL	LADJ, #-1
			19 19 00021	BLSS	1\$
	01		53 D1 00023	CMPL	LADJ, #1
	52		14 14 00026	BGTR	1\$
	51	0C	6E 9E 00028	MOVAB	COLL, R2
			AC D0 0002B	MOVL	CHAR2, R1
	64		64 16 0002F	JSB	COLL VALUE
78			50 E9 00031	BLBC	S, 9\$
			6E B5 00034	TSTW	COLL
			14 12 00036	BNEQ	3\$
			53 D5 00038	TSTL	LADJ
			08 18 0003A	BGEQ	2\$
	50	00000000G	8F D0 0003C	1\$: MOVBL	#COLL\$_ADJ, R0
			04 00043	RET	
			06 15 00044	2\$: BLEQ	3\$
	6E		01 D0 00046	MOVL	#1, COLL
	53		01 CE 00049	MNEG.L	#1, LADJ
	52		6E 9E 0004C	3\$: MOVAB	COLL, R2
	51	08	AC D0 0004F	MOVL	CHAR1, R1
		0000V	30 00053	BSBW	GIVE COLL
	53		50 E9 00056	BLBC	S, 9\$
			53 D5 00059	TSTL	LADJ
			04 12 0005B	BNEQ	4\$
	50		01 D0 0005D	MOVL	#1, R0
			04 00060	RET	
	52	08	6E 9E 00061	4\$: MOVAB	COLL, R2
	51		AC D0 00064	MOVL	CHAR1, R1
			64 16 00068	JSB	COLL_VALUE

3F		50	E9	0006A	BLBC	S, 9\$	0892
50	02	AE	3C	0006D	MOVZWL	COLL+2, S	0898
		03	12	00071	BNEQ	SS	
50		6E	3C	00073	MOVZWL	COLL, S	
		53	D5	00076	TSTL	LADJ	0899
		02	15	00078	BLEQ	6\$	
51		50	D6	0007A	INCL	S	
		50	D0	0007C	MOVL	S, R1	0900
27		0000V	30	0007F	BSBW	D0_BUMP	
		50	E9	00082	BLBC	S,-9\$	0901
52	04	AA	B6	00085	INCW	4(CS)	0902
51	08	6E	9E	00088	MOVAB	COLL, R2	0907
51	08	AC	D0	0008B	MOVL	CHAR1, R1	
		64	16	0008F	JSB	COLL_VALUE	
18		50	E9	00091	BLBC	S, 9\$	0908
		02	AE	00094	TSTW	COLL+2	0913
		06	13	00097	BEQL	7\$	
02	AE	53	A0	00099	ADDW2	LADJ, COLL+2	0914
		03	11	0009D	BRB	8\$	
6E		53	A0	0009F	7\$:	ADDW2	0915
52		6E	9E	000A2	8\$:	LADJ, COLL	
51	08	6E	9E	000A5	MOVAB	COLL, R2	0916
		0000V	30	000A9	MOVL	CHAR1, R1	
		04	000AC	9\$:	BSBW	GIVE_COLL	
					RET		0918

; Routine Size: 173 bytes, Routine Base: SOR\$RO_CODE + 00AF

```
934      0919 1 GLOBAL ROUTINE COLL$FOLD(
935          0920 1     COLL_SEQ:           REF VECTOR[2],
936          0921 1     BV_L:             REF BITVECTOR[K_CHARS], ! The collating sequence
937          0922 1     CC:              ! Lower case letters
938          0923 1     ) =               ! Lower XOR .CC = Upper
939
940          0924 1 ++
941
942          0925 1 FUNCTIONAL DESCRIPTION:
943
944          0928 1 Fold characters (this is a shorthand for several calls to MODIFY).
945          0929 1 For each character (X) in the set of characters specified by BV_L,
946          0930 1 define it to collate equal to its change-case form (X xor CC).
947          0931 1 Also, for all double characters for which neither character is in BV_L,
948          0932 1 define the change-case forms to equal it.
949
950          0934 1 FORMAL PARAMETERS:
951
952          0936 1     COLL_SEQ       a two-longword array specifying the length/address
953          0937 1                 of storage to use for the collating sequence.
954
955          0939 1     BV_L          the address of a 256-bit bitvector.
956
957          0941 1     CC            change-case value to be xor-ed to give the other case.
958
959          0943 1 IMPLICIT INPUTS:
960
961          0945 1 INIT must have already been called.
962
963          0947 1 IMPLICIT OUTPUTS:
964
965          0949 1     NONE
966
967          0951 1 ROUTINE VALUE:
968
969          0953 1     Status code
970
971          0955 1 SIDE EFFECTS:
972
973          0957 1     NONE
974
975          0959 1 --
976          0960 2     BEGIN
977          0961 2     LOCAL
978          0962 2     COLL:   COLL_BLOCK,
979          0963 2     CHAR:  CHAR_BLOCK,
980          0964 2     S:           ! Status value
981          0966 2     CS_SETUP(COLL_SEQ);
982
983          0968 2     ! Define lower case letters to equal their upper case equivalents
984
985          0969 2     CHAR[CHAR_LEN] = 1;
986          0970 2     DECR I FROM K_CHARS-1 TO 0 DO
987          0971 2     IF .BV_L[.I]
988          0972 2     THEN
989          0973 2     BEGIN
990          0974 3     CHAR[CHAR_C0] = .I;
```

```

991      0976 3      S = GIVE_COLL( CHAR[CHAR_ALL], CS[CS_PTAB_.I XOR .CC] );
992      0977 3      IF_ERROR_( .S ) THEN RETURN .S;
993      0978 2      END;
994      0979 2
995      0980 2      ! For all double characters that contain no lower case letters,
996      0981 2      and that contain upper case letters, define lower case forms.
997      0982 2
998      0983 2      CHAR[CHAR_LEN] = 2;
999      0984 4      FOR_ALL_DCHARS(ST)
1000     0985 4          IF NOT .BV_L[.ST[ST_CHAR_0]] AND
1001     0986 4              NOT .BV_L[.ST[ST_CHAR_1]]
1002     0987 4          THEN
1003     0988 5              BEGIN
1004     0989 5                  CHAR[CHAR_C0] = .ST[ST_CHAR];
1005     0990 5                  IF .BV_L[.ST[ST_CHAR_0]] XOR .CC
1006     0991 5                  THEN
1007     0992 6                      BEGIN
1008     0993 6                          CHAR[CHAR_C0] = .CHAR[CHAR_C0] XOR .CC;
1009     0994 6                          S = GIVE_COLL( CHAR[CHAR_A[L]], ST[ST_COLL] );
1010     0995 6                          IF_ERROR_( .S ) THEN RETURN .S;
1011     0996 6                          IF .BV_L[.ST[ST_CHAR_1]] XOR .CC
1012     0997 6                          THEN
1013     0998 7                              BEGIN
1014     0999 7                                  CHAR[CHAR_C1] = .CHAR[CHAR_C1] XOR .CC;
1015     1000 7                                  S = GIVE_COLL( CHAR[CHAR_A[L]], ST[ST_COLL] );
1016     1001 7                                  IF_ERROR_( .S ) THEN RETURN .S;
1017     1002 7                                  CHAR[CHAR_C0] = .CHAR[CHAR_C0] XOR .CC;
1018     1003 7                                  S = GIVE_COLL( CHAR[CHAR_A[L]], ST[ST_COLL] );
1019     1004 7                                  IF_ERROR_( .S ) THEN RETURN .S;
1020     1005 6
1021     1006 6
1022     1007 5          END
1023     1008 5          ELIF .BV_L[.ST[ST_CHAR_1]] XOR .CC
1024     1009 5          THEN
1025     1010 6              BEGIN
1026     1011 6                  CHAR[CHAR_C1] = .CHAR[CHAR_C1] XOR .CC;
1027     1012 6                  S = GIVE_COLL( CHAR[CHAR_A[L]], ST[ST_COLL] );
1028     1013 5                  IF_ERROR_( .S ) THEN RETURN .S;
1029     1014 4
1030     1015 2      END_ALL_DCHARS(ST);
1031     1016 2
1032     1017 2      RETURN SSS_NORMAL;
1033     1      END;

```

	OFFC 00000	.ENTRY	COLL\$FOLD, Save R2,R3,R4,R5,R6,R7,R8,R9,-	: 0919
57	0000V CF 9E 00002	MOVAB	GIVE COLL, R7	
5E	04 C2 00007	SUBL2	#4, SP	: 0966
50	04 AC D0 0000A	MOVL	COLL SEQ, R0	
5A	04 A0 D0 0000E	MOVL	4(R0), CS	: 0970
6E	01 B0 000 2	MOW	#1, CHAR	
55	08 AC D0 00015	MOVL	BV L, R5	: 0972
53	FF 8F 9A 00019	MOVZBL	#255, I	

17		65	53	E1	0001D	1\$:	BBC	I, (R5), 2\$		
		AE	53	90	00021		MOVB	I, CHAR+2	0975	
51	02	53	AC	CD	00025		XORL3	C, I, R1	0976	
		52	0C	CA41	DE	0002A	MOVAL	268(C\$)[R1], R2	:	
		51		6E	9E	00030	MOVAB	CHAR, R1		
				67	16	00033	JSB	GIVE COLL		
		7B		50	E9	00035	BLBC	S, 6\$	0977	
		E2		53	F4	00038	SOBGEQ	I, 1\$	0972	
		6E		02	B0	0003B	MOVW	#2, CHAR	0983	
		53	050C	CA	9E	0003E	MOVAB	1292(R10), ST	0984	
		56	06	AA	3C	00043	MOVZWL	6(C\$), I		
				70	11	00047	BRB	8\$		
66		51		63	9A	00049	MOVZBL	(ST), R1	0985	
		65		51	E0	0004C	BBS	R1, (R5), 7\$		
5E	02	51	01	A3	9A	00050	MOVZBL	1(ST), R1	0986	
		65		51	E0	00054	BBS	R1, (R5), 7\$	0989	
		AE		63	B0	00058	MOVW	(S1), CHAR+2	0990	
		54	0C	AC	D0	0005C	MOVL	CC, R4		
		51		63	9A	00060	MOVZBL	(S1), R1		
31		51		54	CC	00063	XORL2	R4, R1		
		65		51	E1	00066	BBC	R1, (R5), 4\$	0993	
	02	AE		54	8C	0006A	XORB2	R4, CHAR+2	0994	
		52	02	A3	9E	0006E	MOVAB	2(ST), R2		
		51		6E	9E	00072	MOVAB	CHAR, R1		
				67	16	00075	JSB	GIVE COLL		
		45		50	E9	00077	BLBC	S, 9\$	0995	
		51	01	A3	9A	0007A	MOVZBL	1(ST), R1	0996	
31		51		54	CC	0007E	XORL2	R4, R1		
		65		51	E1	00081	BBC	R1, (R5), 7\$	0999	
	03	AE		54	8C	00085	XORB2	R4, CHAR+3	1000	
		52	02	A3	9E	00089	MOVAB	2(ST), R2		
		51		6E	9E	0008D	MOVAB	CHAR, R1		
				67	16	00090	JSB	GIVE COLL		
	02	2A		50	E9	00092	BLBC	S, 9\$	1001	
		AE		54	8C	00095	XORB2	R4, CHAR+2	1002	
				0F	11	00099	BRB	5\$	1003	
10		51	01	A3	9A	0009B	4\$:	MOVZBL	1(ST), R1	1007
		51		54	CC	0009F	XORL2	R4, R1		
		65		51	E1	000A2	BBC	R1, (R5), 7\$	1010	
	03	AE		54	8C	000A6	XORB2	R4, CHAR+3	1011	
		52	02	A3	9E	000AA	5\$::	MOVAB	2(ST), R2	
		51		6E	9E	000AE	MOVAB	CHAR, R1		
				67	16	000B1	JSB	GIVE COLL		
		09		50	E9	000B3	6\$::	BLBC	S, 9\$	1012
		53		06	C0	000B6	7\$::	ADDL2	#6, ST	1015
		8D		56	F4	000B9	8\$::	SOBGEQ	I, 3\$	0984
		50		01	D0	000BC	9\$::	MOVL	#1, R0	1017
				04	000BF		RET		1018	

: Routine Size: 192 bytes, Routine Base: SOR\$RO_CODE + 015C

```
: 1035      1019 1 ROUTINE GIVE_COLL(
: 1036          1020 1     CHAR:  REF CHAR_BLOCK,
: 1037          1021 1     COLL:   REF COLL_BLOCK
: 1038          1022 1     ):    CS_LINK_2 =
: 1039          1023 1     ++
: 1040          1024 1
: 1041          1025 1     FUNCTIONAL DESCRIPTION:
: 1042          1026 1
: 1043          1027 1     Set CHAR to the collating value COLL
: 1044          1028 1
: 1045          1029 1     FORMAL PARAMETERS:
: 1046          1030 1
: 1047          1031 1     CHAR           a character to be defined
: 1048          1032 1
: 1049          1033 1     COLL            a collating value to assign to CHAR
: 1050          1034 1
: 1051          1035 1     IMPLICIT INPUTS:
: 1052          1036 1
: 1053          1037 1     INIT must have already been called.
: 1054          1038 1     CS is specified as a global register.
: 1055          1039 1
: 1056          1040 1     IMPLICIT OUTPUTS:
: 1057          1041 1
: 1058          1042 1     NONE
: 1059          1043 1
: 1060          1044 1     ROUTINE VALUE:
: 1061          1045 1
: 1062          1046 1     Status code
: 1063          1047 1
: 1064          1048 1     SIDE EFFECTS:
: 1065          1049 1
: 1066          1050 1
: 1067          1051 1
: 1068          1052 1     --
: 1069          1053 2     BEGIN
: 1070          1054 2     LOCAL
: 1071          1055 2     TEMP:  REF COLL_BLOCK;
: 1072          1056 2
: 1073          1057 2     CS_SETUP();
: 1074          1058 2
: 1075          1059 2     CASE .CHAR[CHAR_LEN] FROM 1 TO 2 OF
: 1076          1060 2     SET
: 1077          1061 2     [1]:
: 1078          1062 3     BEGIN
: 1079          1063 3     MOVE_COLL_ALL_( CS[CS_PTAB_.CHAR[CHAR_C0]], COLL[COLL_ALL] );
: 1080          1064 2     END;
: 1081          1065 2     [2]:
: 1082          1066 3     BEGIN
: 1083          1067 3     TEMP = D_LOOKUP(.CHAR[CHAR_C01]);
: 1084          1068 3     IF .TEMP EQL 0
: 1085          1069 3     THEN
: 1086          1070 3     TEMP = D_NEW(.CHAR[CHAR_C01]);
: 1087          1071 3     IF .TEMP EQL 0 THEN RETURN COLLS_CMPLX;
: 1088          1072 3     MOVE_COLL_ALL_( TEMP[COLL_ALL], COLL[COLL_ALL] );
: 1089          1073 2     END;
: 1090          1074 2     [INRANGE, OUTRANGE]: RETURN COLLS_CHAR;
: 1091          1075 2     TES;
```

: 1092 1076 2 RETURN SSS_NORMAL:
 : 1093 1077 1 END:

			53 DD 0C000 GIVE_COLL:			
01	0019	0000	51 D0 00002	PUSHL	R3	1019
			63 AF 00005	MOVL	R1, R3	1059
010C CA40	50 00000000G	8F 31 11 00014	00009 1\$: CASEW	(CHAR), #1, #1		
			62 D0 0001A	.WORD	2\$-1\$, - 3\$-1\$	
	51	02 A3 9A 00016	2\$:	MOVL	#COLL\$_CHAR, R0	1074
			62 22 11 00020	BRB	7\$	1063
	51	02 A3 3C 00022	3\$:	MOVZBL	2(CHAR), R0	1059
			0000V 30 00026	MOVL	(COLL), 268(CS)[R0]	1067
	51	02 A3 3C 0002D	50 00029	BRB	6\$	1068
			0000V 30 00031	MOVZWL	2(CHAR), R1	1070
	51	02 A3 3C 00034	07 12 0002B	BSBW	D_LOOKUP	1071
			50 D5 00034	TSTL	TEMP	1072
	50 00000000G	8F 06 11 00038	09 12 00036	BNEQ	4\$	1076
			06 11 0003F	MOVZWL	2(CHAR), R1	1077
	60 50	62 D0 00041	5\$:	BSBW	D_NEW	
			01 D0 00044	TSTL	TEMP	
	60 50	08 BA 00047	6\$::	BNEQ	5\$	
			05 00049	MOVL	#COLL\$_CMPLX, R0	
	60 50	62 D0 00041	7\$::	BRB	7\$	
			01 D0 00044	MOVL	(COLL), (TEMP)	
	60 50	08 BA 00047	7\$::	MOVL	#1, R0	
			05 00049	POPR	#^M<R3>	
				RSB		

: Routine Size: 74 bytes, Routine Base: SOR\$R0_CODE + 021C

```

1095 1078 1 ROUTINE COLL_VALUE(
1096 1079 1     CHAR:           REF CHAR_BLOCK,          ! Character to look up
1097 1080 1     COLL:           REF COLL_BLOCK;        . Collating value (output)
1098 1081 1     ): CS_LINK_2 =
1099 1082 1     ++
1100 1083 1
1101 1084 1     FUNCTIONAL DESCRIPTION:
1102 1085 1
1103 1086 1         Look up the collating value of a character.
1104 1087 1
1105 1088 1     FORMAL PARAMETERS:
1106 1089 1
1107 1090 1         CHAR           a character who's collating value is to be found
1108 1091 1
1109 1092 1         COLL            where CHAR's collating value is to be stored
1110 1093 1
1111 1094 1     IMPLICIT INPUTS:
1112 1095 1
1113 1096 1         INIT must have already been call'ed.
1114 1097 1         CS is specified as a global register.
1115 1098 1
1116 1099 1     IMPLICIT OUTPUTS:
1117 1100 1
1118 1101 1         NONE
1119 1102 1
1120 1103 1     ROUTINE VALUE:
1121 1104 1
1122 1105 1         Status code
1123 1106 1
1124 1107 1     SIDE EFFECTS:
1125 1108 1
1126 1109 1         NONE
1127 1110 1
1128 1111 1     --
1129 1112 2     BEGIN
1130 1113 2     LOCAL
1131 1114 2     TEMP:   COLL_BLOCK,
1132 1115 2     P:      REF COLL_BLOCK;
1133 1116 2
1134 1117 2     CS_SETUP();
1135 1118 2
1136 1119 2     CASE .CHAR[CHAR_LEN] FROM 0 TO 2 OF
1137 1120 2     SET
1138 1121 2
1139 1122 2
1140 1123 3     [2]:
1141 1124 3     BEGIN
1142 1125 3     ! See whether this double character is defined
1143 1126 3
1144 1127 3     P = D_LOOKUP(.CHAR[CHAR_C01]);
1145 1128 3     IF .P NEQ 0
1146 1129 3     THEN
1147 1130 4     BEGIN
1148 1131 4     MOVE COLL_ALL ( COLL(COLL_ALL), P(COLL_ALL) );
1149 1132 4     RETURN SSS_NORMAL;
1150 1133 4
1151 1134 4     END;

```

```

1152      1135 3   ! Take the concatenation of the collating values of
1153      1136 3   the two single characters.
1154      1137 3
1155      1138 3   MOVE_COLL_ALL_( COLL[COLL_ALL], CS[CS_PTAB_(.CHAR[CHAR_0])] );
1156      1139 3   MOVE_COLL_ALL_( TEMP[COLL_ALL], CS[CS_PTAB_(.CHAR[CHAR_1])] );
1157      1140 3   IF .COLL[COLL_C0] EQ0 0
1158      1141 3   THEN
1159      1142 4     MOVE_COLL_ALL_( COLL[COLL_ALL], TEMP[COLL_ALL] )
1160      1143 3   ELIF .COLL[COLL_C1] EQ0 0
1161      1144 3   THEN
1162      1145 4     BEGIN
1163      1146 4       COLL[COLL_C1] = TEMP[COLL_C0];
1164      1147 4       IF .TEMP[COLL_C1] NEQ 0 THEN RETURN COLLS_THREE;
1165      1148 4       END
1166      1149 3   ELSE
1167      1150 3       IF .TEMP[COLL_C0] NEQ 0 THEN RETURN COLLS_THREE;
1168      1151 3   RETURN SSS_NORMAL;
1169      1152 2   END;
1170      1153 2
1171      1154 3
1172      1155 3   [1]: BEGIN
1173      1156 3     MOVE_COLL_ALL_( COLL[COLL_ALL], CS[CS_PTAB_(.CHAR[CHAR_0])] );
1174      1157 2     RETURN SSS_NORMAL;
1175      1158 2   END;
1176      1159 3
1177      1160 3   [0]: BEGIN
1178      1161 3     COLL[COLL_C0] = 0;
1179      1162 3     COLL[COLL_C1] = 0;
1180      1163 2     RETURN SSS_NORMAL;
1181      1164 2   END;
1182      1165 2   [INRANGE, OUTRANGE]:
1183      1166 2     RETURN COLLS_CMPLX;
1184      1167 2
1185      1168 2   TES:
1186      1169 1   END;

```

			53 DD 00000 COLL_VALUE:			
			PUSHL R3			1078
		02	04 C2 00002	SUBL2 #4, SP		
		000F	53 00 0005	MOVL R1, R3		
		000F	63 AF 00008	CASEW (CHAR), #0, #2		1159
		000F	0000C 1\$: .WORD	8S-1\$, -		
		000F		7S-1\$, -		
		000F		2S-1\$		
		000F	50 0000000G 8F D0 00012	MOVL #COLL\$_CMPLX, R0		1165
		000F	59 11 00019	BRB 10\$		
		000F	51 02 A3 3C 0001B 2\$: 0000V 30 0001F	MOVZWL 2(CHAR), R1		1127
		000F	50 D5 00022	BSBW D_LOOKUP		
		000F	05 13 00024	TSTL P		1128
		000F	62 60 D0 00026	BEQL 3\$		
		000F	46 11 00029	MOVL (P), (COLL)		1131
		000F	50 02 A3 9A 0002B 3\$:	BRB 9\$		1132
		000F		MOVZBL 2(CHAR), R0		1138

62	010C	CA40	D0	0002F	MOVL	268((CS)[R0], (COLL))	
50	03	A3	9A	00035	MOVZBL	3(CHAR) R0	1139
6E	010C	CA40	D0	00039	MOVL	268((CS)[R0], TEMP)	
			62	B5 0003F	TSTW	(COLL)	1140
			05	12 00041	BNEQ	4\$	
62			6E	D0 00043	MOVL	TEMP, (COLL)	1142
			29	11 00046	BRB	9\$	
	02	A2	85 00048	4\$:	TSTW	2(COLL)	1140
			09	12 0004B	BNEQ	5\$	1143
02	A2		6E	80 0004D	MOVW	TEMP, 2(COLL)	
			02	AE B5 00051	TSTW	TEMP+2	1146
			02	11 00054	BRB	6\$	1147
			6E	B5 00056	TSTW	TEMP	
			17	13 00058	BEQL	9\$	1150
50	000000006		8F	D0 0005A	MOVL	#COLL\$_THREE, R0	
			11	11 00061	BRB	10\$	
50	02	A3	9A 00063	7\$:	MOVZBL	2(CHAR), R0	1155
62	010C	CA40	D0	00067	MOVL	268((CS)[R0], (COLL))	
			02	11 0006D	BRB	9\$	1159
50			62	D4 0006F	CLRL	(COLL)	
	01		01	D0 00071	MOVL	#1, R0	1160
5E			04	C0 00074	ADDL2	#4, SP	
			08	BA 00077	POPR	#^M<R3>	1162
			05	00079	RSB		1169

: Routine Size: 122 bytes, Routine Base: SOR\$R0_CODE + 0266

```
1188      1170 1 GLOBAL ROUTINE COLL$TIE_BREAK(
1189          1171 1     COLL_SEQ:    REF VECTOR[2],
1190          1172 1     ORDER
1191          1173 1     ) =
1192          1174 1     ++
1193          1175 1
1194          1176 1     FUNCTIONAL DESCRIPTION:
1195          1177 1
1196          1178 1     Indicates that a tie-breaking comparison should be done to distinguish
1197          1179 1     records that compare equal with the primary and secondary comparisons.
1198          1180 1     This tie-breaking comparison is a simple binary string compare.
1199          1181 1
1200          1182 1     FORMAL PARAMETERS:
1201          1183 1
1202          1184 1     COLL_SEQ      a two-longword array specifying the length/address
1203          1185 1     of storage to use for the collating sequence.
1204          1186 1
1205          1187 1     ORDER         indicates whether the simple comparison part of the
1206          1188 1     tie-breaking should be:
1207          1189 1     In the normal order   (ORDER = FALSE) or
1208          1190 1     In the opposite order (ORDER = TRUE).
1209          1191 1
1210          1192 1     This distinction is important for DEC STD 169, which places lower case
1211          1193 1     letters before their upper case equivalents. It is unrelated to whether
1212          1194 1     the keys are ascending or descending.
1213          1195 1
1214          1196 1
1215          1197 1     IMPLICIT INPUTS:
1216          1198 1
1217          1199 1     INIT must have already been called.
1218          1200 1
1219          1201 1     IMPLICIT OUTPUTS:
1220          1202 1
1221          1203 1     NONE
1222          1204 1
1223          1205 1
1224          1206 1
1225          1207 1
1226          1208 1
1227          1209 1
1228          1210 1
1229          1211 1
1230          1212 1
1231          1213 1     SIDE EFFECTS:
1232          1214 2
1233          1215 2
1234          1216 2
1235          1217 2
1236          1218 2
1237          1219 2
1238          1220 2
1239          1221 2
1240          1222 2
1241          1223 2
1242          1224 2
1243          1225 1
1232          -- BEGIN
1233          1214 2
1234          1215 2
1235          1216 2
1236          1217 2
1237          1218 2
1238          1219 2
1239          1220 2
1240          1221 2
1241          1222 2
1242          1223 2
1243          1224 2
1244          1225 1
1232          CS_SETUP(COLL_SEQ);
1233          CS[CS_TB] = .CS[CS_TB] AND NOT TBSNOTB;
1234          IF .ORDER
1235          THEN
1236              CS[CS_REVERSE] = .CS[CS_REVERSE] OR TBSREVERSE;
1237          RETURN SSS_NORMAL;
1238          END;
```

				.ENTRY	COLL\$TIE_BREAK, >a,e R10	:	1170
	50	04	0400 00000	MOVL	COLL SEQ, R0	:	1216
	5A	04	AC DD 00002	MOVL	4(ROT), CS	:	
08	AA	04	A0 DD 00006	BICB2	#4, 8(CS)	:	1218
	04	08	04 8A 0000A	BLBC	ORDER 1\$:	1220
0A	AA	08	AC E9 0000E	BISB2	#1, 10(CS)	:	1222
	50	01	01 88 00012	MOVL	#1, R0	:	1224
		01	DD 00016	1\$:	RET	:	1225
		04	00019				

: Routine Size: 26 bytes, Routine Base: SOR\$R0_CODE + 02E0

```
: 1245      1226 1 GLOBAL ROUTINE COLL$PAD(
: 1246      1227 1   COLL_SEQ:    REF VECTOR[2],
: 1247      1228 1   PAD:        REF CHAR_BLOCK
: 1248      1229 1   ) =
: 1249      1230 1   ++
: 1250      1231 1
: 1251      1232 1   FUNCTIONAL DESCRIPTION:
: 1252      1233 1
: 1253      1234 1   Specifies a pad character to be used in comparisons of strings with
: 1254      1235 1   different lengths.
: 1255      1236 1
: 1256      1237 1   FORMAL PARAMETERS:
: 1257      1238 1
: 1258      1239 1   COLL_SEQ      a two-longword array specifying the length/address
: 1259      1240 1   of storage to use for the collating sequence.
: 1260      1241 1
: 1261      1242 1   PAD          the pad character.
: 1262      1243 1
: 1263      1244 1   IMPLICIT INPUTS:
: 1264      1245 1
: 1265      1246 1   INIT must have already been called.
: 1266      1247 1
: 1267      1248 1   IMPLICIT OUTPUTS:
: 1268      1249 1
: 1269      1250 1   NONE
: 1270      1251 1
: 1271      1252 1   ROUTINE VALUE:
: 1272      1253 1
: 1273      1254 1   Status code
: 1274      1255 1
: 1275      1256 1   SIDE EFFECTS:
: 1276      1257 1
: 1277      1258 1   NONE
: 1278      1259 1
: 1279      1260 1   NOTES:
: 1280      1261 1
: 1281      1262 1   Assertion:
: 1282      1263 1   The purpose of a pad character is to allow strings of different
: 1283      1264 1   lengths to compare equal.
: 1284      1265 1   Therefore:
: 1285      1266 1   There is no reason to have a different pad character for the
: 1286      1267 1   tie-break.
: 1287      1268 1   Also:
: 1288      1269 1   It is unreasonable for the pad character to be ignored.
: 1289      1270 1   It is unreasonable for the pad character to be the second
: 1290      1271 1   character of a double character.
: 1291      1272 1
: 1292      1273 1   -
: 1293      1274 2   BEGIN
: 1294      1275 2
: 1295      1276 2   CS_SETUP(COLL_SEQ);
: 1296      1277 2
: 1297      1278 2   IF .PAD[CHAR_LEN] NEQ 1 THEN RETURN COLLS_PAD;
: 1298      1279 2   CS[CS_PAD] = .PAD[CHAR_C0];
: 1299      1280 2   RETURN SSS_NORMAL;
: 1300      1281 2
: 1301      1282 1   END;
```

				0400 00000	.ENTRY	COLL\$PAD, Save R10	:	1226	
50	04	AC	DO	00002	MOVL	COLL SEQ, R0	:	1276	
5A	04	A0	DO	00006	MOVL	4(R0), CS	:		
50	08	AC	DO	0000A	MOVL	PAD, R0	:	1278	
01		60	B1	0000E	CMPW	(R0), #1	:		
		08	13	00011	BEQL	1\$:		
50	00000000G	8F	DO	00013	MOVL	#COLL\$_PAD, R0	:		
			04	0001A	RET		:		
09	AA	02	A0	90 0001B	1\$:	MOVB	2(R0), 9(CS)	:	1279
			01	DO 00020		MOVL	#1, R0	:	1280
				04 00023		RET		1281	

; Routine Size: 36 bytes, Routine Base: SOR\$RO_CODE + 02FA

```

1303      1283 1 ROUTINE DO_BUMP(X: WORD): CS_LINK_1 =           ! Bump collating values >= X
1304      1284 1
1305      1285 1   +
1306      1286 1
1307      1287 1 FUNCTIONAL DESCRIPTION:
1308      1288 1
1309      1289 1   Create an unused collating value by increasing all collating values
1310      1290 1   that are greater than or equal to X.
1311      1291 1
1312      1292 1 FORMAL PARAMETERS:
1313      1293 1
1314      1294 1   X           a (single) collating value, passed as a word.
1315      1295 1
1316      1296 1 IMPLICIT INPUTS:
1317      1297 1
1318      1298 1   INIT must have already been called.
1319      1299 1   CS is specified as a global register.
1320      1300 1
1321      1301 1 IMPLICIT OUTPUTS:
1322      1302 1   NONE
1323      1303 1
1324      1304 1
1325      1305 1 ROUTINE VALUE:
1326      1306 1
1327      1307 1   Status code
1328      1308 1
1329      1309 1 SIDE EFFECTS:
1330      1310 1
1331      1311 1   NONE
1332      1312 1
1333      1313 1 --!
1334      1314 2 BEGIN
1335      1315 2 MACRO
1336      1316 2   BUMP_(Z) = IF .Z GEQ .X THEN Z = .Z + 1 ELSE 0 %;
1337      1317 2
1338      1318 2
1339      1319 2
1340      1320 5 FOR_ALL_COLL(S(P)
1341      1321 5   BUMP_(P[COLL_C0]);
1342      1322 5   BUMP_(P[COLL_C1]);
1343      1323 2 END_ALL_COLL(S(P);
1344      1324 2 RETURN $SS_NORMAL;
1345      1325 1 END;

```

52	010C	1C BB 00000 DO_BUMP: PUSHR	#^M<R2,R3,R4>	: 1283
54		CA 9E 00002	MOVAB 268(R10), P	: 1320
53	04	DO 00007	MOVL #4, STEP	: 1321
07	01	DO 0000A	MOVL #1, FIRST	: 1320
50	0100	53 E9 0000D 1\$: BLBC FIRST, 2\$: 1320	
50	06	8F 3C 00010	MOVZWL #256, R0	: 1321
		19 11 00015	BRB 6\$: 1323
		AA 3C 00017 2\$: MOVZWL 6(CS) R0	: 1323	
		13 11 0001B	BRB 6\$	

51		62 B1 0001D 3\$:	CMPW (P), X	: 1321
		02 1F 00020	BLSSU 4\$	
		62 B6 00022	INCW (P)	
51	02	A2 B1 00024 4\$:	CMPW 2(P), X	: 1322
		03 1F 00028	BLSSU 5\$	
	02	A2 B6 0002A	INCW 2(P)	
52		54 C0 0002D 5\$:	ADDL2 STEP, P	: 1323
EA		50 F4 00030 6\$:	S0BGEQ I, 3\$: 1320
54		06 D0 00033	MOVL #6, STEP	: 1323
52		02 C0 00036	ADDL2 #2, P	
D1		53 F4 00039	S0BGEQ FIRST, 1\$	
50		01 D0 0003C	MOVL #1, R0	: 1320
		1C BA 0003F	POPR #^M<R2,R3,R4>	: 1324
		05 00041	RSB	: 1325

: Routine Size: 66 bytes, Routine Base: SOR\$RO_CODE + 031E

```

1347    1326 1 ROUTINE D_NEW(X: WORD): CS_LINK_1 =           ! Get space for new double char
1348    1327 1
1349    1328 1 ++
1350    1329 1
1351    1330 1 FUNCTIONAL DESCRIPTION:
1352    1331 1
1353    1332 1     Get space for the new double character specified by X, and return the
1354    1333 1     address in which its collating value will be stored.
1355    1334 1
1356    1335 1 FORMAL PARAMETERS:
1357    1336 1
1358    1337 1     X          a (double) character, passed as a word.
1359    1338 1
1360    1339 1 IMPLICIT INPUTS:
1361    1340 1
1362    1341 1     INIT must have already been called.
1363    1342 1     CS is specified as a global register.
1364    1343 1
1365    1344 1 IMPLICIT OUTPUTS:
1366    1345 1
1367    1346 1     NONE
1368    1347 1
1369    1348 1 ROUTINE VALUE:
1370    1349 1
1371    1350 1     The address in which the collating value will be stored,
1372    1351 1     Or zero if no more space is available.
1373    1352 1
1374    1353 1 SIDE EFFECTS:
1375    1354 1
1376    1355 1     NONE
1377    1356 1
1378    1357 1 -- BEGIN
1379    1358 2 LOCAL
1380    1359 2
1381    1360 2     P: REF ST_BLOCK;
1382    1361 2     CS_SETUP();
1383    1362 2     P = .CS[CS_CURR_SIZE] + ST_K_SIZE;
1384    1363 2     IF .P GTTU .CS[CS_SIZE] THEN RETURN 0;      ! No more storage!
1385    1364 2     CS[CS_CURR_SIZE] = .P;
1386    1365 2     CS[CS_DCHAR] = .CS[CS_DCHAR] + 1;
1387    1366 2     P = .P + CS[BASE_] - ST_K_SIZE;
1388    1367 2     P[ST_CHAR] = .X;
1389    1368 2     RETURN P[ST_COLL];
1390    1369 1 END;

```

		50	02	AA	3C 00000 D_NEW:	MOVZWL 2(CS), P	: 1362
50	6A	50	06	C0 00004	ADDL2 #6, P	: 1363	
		10	00	ED 00007	CMPZV #0, #16, (CS), P	: 1364	
			10	1F 0000C	BLSSU 1\$: 1365	
		02	AA	50 B0 0000E	MOVW P, 2(CS)	: 1366	
			06	AA B6 00012	INCW 6(CS)	: 1367	
		50	FA AA40	9E 00015	MOVAB -6(CS)[P], P		
		80		51 B0 0001A	MOVW X, (P)+		

COLL\$UTILITIES
V04-000

F 14

16-Sep-1984 01:06:02
14-Sep-1984 13:10:40

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SORCOLUTI.B32;1

Page 44
(22)

50 05 0001D RSB
D4 0001E 1\$: CLRL R0
05 00020 RSB

: 1368
: 1369

; Routine Size: 33 bytes. Routine Base: SCR\$R0_CODE + 0360

```

1392    1370 1 ROUTINE D_LOOKUP(X: WORD): CS_LINK_1 =      ! Look up a double character
1393    1371 1
1394    1372 1 ++
1395    1373 1
1396    1374 1 FUNCTIONAL DESCRIPTION:
1397    1375 1
1398    1376 1     Get the collating value for a double character.
1399    1377 1
1400    1378 1 FORMAL PARAMETERS:
1401    1379 1
1402    1380 1     X           a (double) character, passed as a word.
1403    1381 1
1404    1382 1 IMPLICIT INPUTS:
1405    1383 1
1406    1384 1     INIT must have already been called.
1407    1385 1     CS is specified as a global register.
1408    1386 1
1409    1387 1 IMPLICIT OUTPUTS:
1410    1388 1
1411    1389 1     NONE
1412    1390 1
1413    1391 1 ROUTINE VALUE:
1414    1392 1
1415    1393 1     The address of the collating value,
1416    1394 1     Or zero if the double character is undefined.
1417    1395 1
1418    1396 1 SIDE EFFECTS:
1419    1397 1
1420    1398 1     NONE
1421    1399 1
1422    1400 1 ---
1423    1401 2 BEGIN
1424    1402 2
1425    1403 2     CS_SETUP();
1426    1404 2
1427    1405 4 FOR_ALL_DCHARS(ST)
1428    1406 4     IF .ST[ST_CHAR] EQL .X THEN RETURN ST[ST_COLL];
1429    1407 2 END_ALL_DCHARS(ST);
1430    1408 2
1431    1409 2 RETURN 0;
1432    1410 1 END;

```

			OC	BB 00000 D_LOOKUP:			
50	050C	06	CA	9E 00002	PUSHR	#^M<R2,R3>	: 1370
53			AA	3C 00007	MOVAB	1292(R10), ST	: 1405
				11 11 0000B	MOVZWL	6(CS), I	: 1406
51			60	B1 0000D 1\$:	BRB	3\$	
			60	09 12 00010	CMPW	(ST), X	
52	02		A0	9E 00012	BNEQ	2\$	
50			52	D0 00016	MOVAB	2(R0), R2	
			50	08 11 00019	MOVL	R2, R0	
50		06	C0	0001B 2\$: ADDL2	BRB	4\$	
					ADDL2	#6, ST	: 1407

COLL\$UTILITIES
V04-000

H 14
16-Sep-1984 01:06:02 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:40 [SORT32.SRC]SORCOLUTI.B32;1

Page 46
(23)

EC 53 F4 0001E 3\$: SOBGEQ I 1\$
50 D4 00021 CLRL R0
0C BA 00023 4\$: POPR #^M<R2,R3>
05 00025 RSB

; 1405
; 1409
; 1410
;

: Routine Size: 38 bytes, Routine Base: SOR\$RO_CODE + 0381

```
1434      1411 1 | To compress the range of collating values, we must determine what values
1435      1412 1 | are currently in use. In practice, we will determine which of the values
1436      1413 1 | 1..MAX_USED-1 are in use, and whether any larger values are in use.
1437      1414 1 | Unused values are freed, and larger values are decreased; repeat as needed.
1438      1415 1 | If more than 2^K CHARS distinct values are in use, it would be almost
1439      1416 1 | impossible to "double-up" sufficient values to fit things in a byte, and
1440      1417 1 | certainly not by this code.
1441      1418 1 |
1442      1419 1 | LITERAL
1443      1420 1 |   MAX_USED = 2 * K_CHARS;
1444      1421 1 |
1445      1422 1 | ROUTINE COMPRESS
1446      1423 1 | (
1447      1424 1 |   S_BV: REF BITVECTOR[MAX_USED]
1448      1425 1 | ): CS_LINK_1 =
1449      1426 1 | ++
1450      1427 1 |
1451      1428 1 | FUNCTIONAL DESCRIPTION:
1452      1429 1 |
1453      1430 1 | Reduce the range of collating values in use by simply accounting for
1454      1431 1 | unused collating values.
1455      1432 1 |
1456      1433 1 | FORMAL PARAMETERS:
1457      1434 1 |
1458      1435 1 | S_BV          bitvector (output parameter)
1459      1436 1 | each bit indicates whether the corresponding collating
1460      1437 1 | value is in use.
1461      1438 1 |
1462      1439 1 | IMPLICIT INPUTS:
1463      1440 1 |
1464      1441 1 | INIT must have already been called.
1465      1442 1 | CS is specified as a global register.
1466      1443 1 |
1467      1444 1 | IMPLICIT OUTPUTS:
1468      1445 1 |
1469      1446 1 |   NONE
1470      1447 1 |
1471      1448 1 | ROUTINE VALUE:
1472      1449 1 |
1473      1450 1 |   Status code
1474      1451 1 |
1475      1452 1 | SIDE EFFECTS:
1476      1453 1 |
1477      1454 1 |   NONE
1478      1455 1 |
1479      1456 1 | --
1480      1457 2 | BEGIN
1481      1458 2 | LOCAL
1482      1459 2 |   S_BV_0,
1483      1460 2 |   USED: XBLISS16(REF) VECTOR[MAX_USED,WORD],
1484      1461 2 |   FREED:
1485      1462 2 |
1486      1463 2 | MACRO SET_BV_(VAL) = IF .VAL LSSU MAX_USED THEN S_BV[.VAL] = TRUE ELSE 0 %;
1487      1464 2 | MACRO DEC_BV_(VAL) = IF .VAL LSSU MAX_USED THEN VAL = .USED[.VAL]
1488      1465 2 |                   ELSE (VAL = .VAL = .FREED; S_BV_0 = FALSE) %;
1489      1466 2 |
1490      1467 2 | CS_SETUP();
```

```
: 1491      1468 2
: 1492      1469 2
: 1493      1470 2
: 1494      U 1471 2
: 1495      U 1472 2
: 1496      U 1473 2
: 1497      U 1474 2
: 1498      1475 2
: 1499      1476 2
: 1500      1477 2
: 1501      1478 2
: 1502      1479 2
: 1503      1480 2
: 1504      1481 2
: 1505      1482 2
: 1506      1483 3
: 1507      1484 3
: 1508      1485 3
: 1509      1486 3
: 1510      1487 3
: 1511      1488 3
: 1512      1489 3
: 1513      1490 3
: 1514      1491 6
: 1515      1492 6
: 1516      1493 6
: 1517      1494 3
: 1518      1495 3
: 1519      1496 3
: 1520      1497 3
: 1521      1498 3
: 1522      1499 3
: 1523      1500 3
: 1524      1501 4
: 1525      1502 4
: 1526      1503 4
: 1527      1504 4
: 1528      1505 3
: 1529      1506 3
: 1530      1507 3
: 1531      1508 3
: 1532      1509 3
: 1533      1510 6
: 1534      1511 6
: 1535      1512 6
: 1536      1513 3
: 1537      1514 3
: 1538      1515 3
: 1539      1516 3
: 1540      1517 3
: 1541      1518 3
: 1542      1519 3
: 1543      1520 3
: 1544      1521 2
: 1545      1522 2
: 1546      1523 2
: 1547      1524 1

        ! Allocate the USED vector in the work area
        %IF %BLISS(BLISS16) %THEN
        USED = .CS[CS_CURR_SIZE] + %SIZE(VECTOR[MAX_USED,WORD]);
        IF .USED GTRU .CS[CS_SIZE] THEN RETURN COLLS_CMPLX;
        USED = .USED + [CS$BASE_] - %SIZE(VECTOR[MAX_USED,WORD]);
        %FI

        ! Use as few distinct collating values as possible
        ! (without converting single to double or double to single).
        USED[0] = 0;
        WHILE TRUE DO
            BEGIN
                LOCAL
                    VAL;

            ! Determine which old collating values are being used
            DECR I FROM MAX_USED/%BPVAL-1 TO 0 DO (S_BV[0]+(.I*%UPVAL)) = 0;
            %IF (MAX_USED - MAX_USED/%BPVAL*%BPVAL) NEQ 0 %THEN %ERROR('') %FI
            FOR_ALL_COLL(p)
                SET_BV_(P[COLL_C0]);
                SET_BV_(P[COLL_C1]);
            END_ALL_COLL(p);

            ! Compute the new collating values
            FREED = 0;
            VAL = 0;
            INCR I FROM 1 TO MINU(MAX_USED-1, .CS[CS_COLL_MAX]) DO
                BEGIN
                    IF .S_BV[I]
                        THEN USED[I] = VAL = .VAL + 1
                    ELSE FREED = .FREED + 1;
                END;

            ! Now convert to the new collating values
            S_BV_0 = TRUE;
            FOR_ALL_COLL(p)
                DEC_BV_(P[COLL_C0]);
                DEC_BV_(P[COLL_C1]);
            END_ALL_COLL(p);
            CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] - .FREED;

            ! Continue?
            IF .S_BV_0 THEN EXITLOOP;
            IF .FREED EQL 0 THEN RETURN COLLS_CMPLX;           ! We got everything
                                                               ! Can't do any more
            END;

        RETURN SSS_NORMAL;
        END;
```

		01FC	8F	BB 00000 COMPRESS:		
5E	FC00	CE 9E 00004		PUSHR #^M<R2,R3,R4,R5,R6,R7,R8>		1422
58	010C	6E B4 00009		MOVAB -1024(SP), SP		1481
50		CA 9E 0000B		CLRW USED		1491
		OF D0 00010	1\$:	MOVAB 268(R10), R8		1489
		6140 D4 00013	2\$:	MOVL #15, I		
FA		50 F4 00016		CLRL (S_BV)[I]		
52		58 D0 00019		SOBGEQ I-2\$		
54		04 D0 0001C		MOVL R8, P		1491
53		01 D0 0001F		MOVL #4, STEP		
07		53 E9 00022	3\$:	MOVL #1, FIRST		
50	0100	8F 3C 00025		BLBC FIRST, 4\$		
		27 11 0002A		MOVZWL #256, R0		
50	06	AA 3C 0002C	4\$:	BRB 8\$		
		21 11 00030		MOVZWL 6(CS), R0		1494
0200	8F	62 B1 00032	5\$:	BRB 8\$		1492
		07 1E 00037		CMPW (P), #512		
00	55	62 3C 00039		BGEQU 6\$		
00	0200	61 A2 B1 00040	6\$:	MOVZWL (P), R5		
		08 1F 00046		BBSS R5, (S_BV), 6\$		1493
00	55	A2 3C 00048		BGEQU 7\$		
	61	55 E2 0004C		MOVZWL 2(P), R5		
52		54 C0 00050	7\$:	BBSS R5, (S_BV), 7\$		1494
DC		50 F4 00053	8\$:	ADDL2 STEP, P		1491
54		06 D0 00056		SOBGEQ I, 5\$		1494
52		02 C0 00059		MOVL #6, STEP		
C3		53 F4 0005C		ADDL2 #2, P		1491
		56 D4 0005F		SOBGEQ FIRST, 3\$		1498
		53 D4 00061		CLRL FREED		1499
01FF	52	04 AA 3C 00063		CLRL VAL		
	8F	52 B1 00067		MOVZWL 4(CS), R2		1500
	52	05 18 0006C		CMPW R2, #511		
	01FF	8F 3C 0006E		BLEQU 9\$		
		50 D4 00073	9\$:	MOVZWL #511, R2		
08	61	0E 11 00075		CLRL I		
		50 E1 00077	10\$:	BRB 12\$		1502
	6E40	53 D6 0007B		BBC I, (S_BV), 11\$		1503
		53 B0 0007D		INCL VAL		
EE	50	02 11 00081		MOVW VAL, USED[I]		
		56 D6 00083	11\$:	BRB 12\$		
	57	52 F3 00085	12\$:	INCL FREED		1504
52		01 D0 00089		AOBLEQ R2, I, 10\$		1500
55		58 D0 0008C		MOVL #1, S_BV_0		1509
54		04 D0 0008F		MOVL R8, P		1510
07		01 D0 00092		MOVL #4, STEP		
50	0100	54 E9 00095	13\$:	MOVL #1, FIRST		
		8F 3C 00098		BLBC FIRST, 14\$		
50	06	36 11 0009D		MOVZWL #256, R0		
		AA 3C 0009F	14\$:	BRB 20\$		
		30 11 000A3		MOVZWL 6(CS), R0		1513
				BRB 20\$		

0200	53	62	3C 000A5	15\$:	MOVZWL (P), R3		1511
	8F	53	B1 000A8		CMPW R3, #512		
		06	1E 000AD		BGEQU 16\$		
		62	6E43 B0 000AF		MOVW USED[R3], (P)		
			05 11 000B3		BRB 17\$		
		62	56 A2 000B5	16\$:	SUBW2 FREED, (P)		
			57 D4 000B8		CLRL S BV-0		
0200	53	02	A2 3C 000BA	17\$:	MOVZWL 27P), R3		1512
	8F		53 B1 000BE		CMPW R3, #512		
		02	07 1E 000C3		BGEQU 18\$		
		02	A2 6E43 B0 000C5		MOVW USED[R3], 2(P)		
			06 11 000CA		BRB 19\$		
		02	A2 56 A2 000CC	18\$:	SUBW2 FREED, 2(P)		
			57 D4 000D0		CLRL S BV-0		
		52	55 C0 000D2	19\$:	ADDL2 STEP- P		1513
		CD	50 F4 000D5	20\$:	SOBGEQ I, 15\$		1510
		55	06 D0 000D8		MOVL #6, STEP		1513
		52	02 C0 000DB		ADDL2 #2, P		
		B4	54 F4 000DE		SOBGEQ FIRST, 13\$		1510
04	AA		56 A2 000E1		SUBW2 FREED, 4(CS)		1514
	10		57 E8 000E5		BLBS S BV 0, 22\$		1518
			56 D5 000E8		TSTL FREED		1519
			03 13 000EA		BEQL 21\$		
			FF21 31 000EC		BRW 1\$		
50	00000000G	8F	D0 000EF	21\$:	MOVL #COLL\$_CMPLX, R0		
			03 11 000F6		BRB 23\$		
50		01	D0 000F8	22\$:	MOVL #1, R0		1523
5E	0400	CE	9E 000FB	23\$:	MOVAB 1024(SP), SP		1524
	01FC	8F	BA 00100		POPR #^M<R2,R3,R4,R5,R6,R7,R8>		
			05 00104		RSB		

: Routine Size: 261 bytes, Routine Base: SOR\$R0_CODE + 03A7

```
: 1549      1525 1 ROUTINE COMPRESS_M: CS_LINK_0 =
: 1550      1526 1
: 1551      1527 1 ++
: 1552      1528 1
: 1553      1529 1 FUNCTIONAL DESCRIPTION:
: 1554      1530 1
: 1555      1531 1 Convert the collating sequence to the efficient and succinct form that's
: 1556      1532 1 used by the comparison routines.
: 1557      1533 1
: 1558      1534 1 FORMAL PARAMETERS:
: 1559      1535 1
: 1560      1536 1     NONE
: 1561      1537 1
: 1562      1538 1 IMPLICIT INPUTS:
: 1563      1539 1
: 1564      1540 1 INIT must have already been called.
: 1565      1541 1 CS is specified as a global register.
: 1566      1542 1
: 1567      1543 1 IMPLICIT OUTPUTS:
: 1568      1544 1
: 1569      1545 1     NONE
: 1570      1546 1
: 1571      1547 1 ROUTINE VALUE:
: 1572      1548 1
: 1573      1549 1     Status code
: 1574      1550 1
: 1575      1551 1 SIDE EFFECTS:
: 1576      1552 1
: 1577      1553 1     NONE
: 1578      1554 1
: 1579      1555 1 --
: 1580      1556 2 BEGIN
: 1581      1557 2 LOCAL
: 1582      1558 2     BV:     BITVECTOR[MAX_USED].
: 1583      1559 2     NEED,
: 1584      1560 2     S;                      ! Status value
: 1585      1561 2
: 1586      1562 2     CS_SETUP();
: 1587      1563 2
: 1588      1564 2
: 1589      1565 2     We are going to mash this collating sequence down to size.
: 1590      1566 2
: 1591      1567 2
: 1592      1568 2     First, check that the pad character isn't used in any double characters.
: 1593      1569 2     and that it collates to a single byte collating value.
: 1594      1570 2
: 1595      1571 4     FOR_ALL_DCHARS(ST)
: 1596      1572 4         IF .ST[ST_CHAR_0] EQL .CS[CS_PAD]
: 1597      1573 4         OR .ST[ST_CHAR_1] EQL .CS[CS_PAD] THEN RETURN COLLS_PAD;
: 1598      1574 2     END_ALL_DCHARS(ST);
: 1599      1575 2     IF .BBLOCK[CS[CS_P1AB_(.CS[CS_PAD])],COLL_C1] NEQ 0 THEN RETURN COLLS_PAD;
: 1600      1576 2
: 1601      1577 2
: 1602      1578 2     Use fewer collating values
: 1603      1579 2
: 1604      1580 2     S = COMPRESS(BV[0]);
: 1605      1581 2     IF_ERROR_( .S ) THEN RETURN .S;
```

```
: 1606      1582  2
: 1607      1583  2
: 1608      1584  2
: 1609      1585  2
: 1610      1586  2
: 1611      1587  2
: 1612      1588  5
: 1613      1589  5
: 1614      1590  5
: 1615      1591  2
: 1616      1592  2
: 1617      1593  2
: 1618      1594  2
: 1619      1595  2
: 1620      1596  2
: 1621      1597  2
: 1622      1598  2
: 1623      1599  2
: 1624      1600  2
: 1625      1601  2
: 1626      1602  2
: 1627      1603  2
: 1628      1604  2
: 1629      1605  2
: 1630      1606  2
: 1631      1607  2
: 1632      1608  2
: 1633      1609  2
: 1634      1610  2
: 1635      1611  2
: 1636      1612  2
: 1637      1613  2
: 1638      1614  2
: 1639      1615  2
: 1640      1616  2
: 1641      1617  2
: 1642      1618  2
: 1643      1619  2
: 1644      1620  3
: 1645      1621  3
: 1646      1622  3
: 1647      1623  3
: 1648      1624  3
: 1649      1625  3
: 1650      1626  3
: 1651      1627  3
: 1652      1628  3
: 1653      1629  3
: 1654      1630  3
: 1655      1631  3
: 1656      1632  6
: 1657      1633  6
: 1658      1634  6
: 1659      1635  7
: 1660      1636  7
: 1661      1637  7
: 1662      1638  6

| Determine some attributes.
| Are there any ignored collating values.
| Are there any double collating values.

FOR_ALL_COLLSS(P)
  IF .P[COLL_0] EQL 0 THEN CS[CS_IGN] = TRUE;
    IF .P[COLL_1] NEQ 0 THEN CS[CS_DCOLL] = TRUE;
END_ALL_COLLSS(P);

| A double character <i0,i1> with double collating value <c0,c1> can
| be deleted if:
|   The collating value of <i0,0> is <c0,0>, and
|   The collating value of <i1,0> is <c1,0>, and
|   There are no double characters of the form: <i1,z>

0:

| Determine whether to convert single collating values to double,
| Or to convert double to single.

NEED = .CS[CS_COLL_MAX] - K_CHARS;

| If we already have double collating values or double characters,
| there's not much harm in creating one more to create a free collating
| value. This is advantageous in the comparison routine; also necessary,
| since 0 will be used to indicate a special character.

IF .CS[CS_DCOLL] OR .CS[CS_DCHAR] GTR 0 THEN NEED = .NEED + 1;

| Recall that, on entry to this block, bv[x] indicates that
| the collating value is in use.

IF .NEED GTR 0
THEN
  BEGIN
    | Convert single to double
    | Find a sequence of adjacent (single) collating values that
    | are not used in a double collating value.
    | Convert characters with these collating values to have double
    | collating values.

    LOCAL
      CHAR: CHAR_BLOCK,
      S, Q;
    FOR_ALL_COLLSS(P)
      IF .P[COLL_1] NEQ 0
      THEN
        BEGIN
          BV[.P[COLL_0]] = FALSE;
          BV[.P[COLL_1]] = FALSE;
        END;
```

```
1663      1639 3      END_ALL_COLLS(P);
1664      1640 3      BV[BBLLOCK[CSECS_PTAB_(CS[CS_PAD])],COLL_C0]] = FALSE;
1665      1641 3      BV[0] = FALSE;
1666      1642 3
1667      1643 3      | Now we know what single collating values are available
1668      1644 3
1669      1645 3      CHAR[CHAR_LEN] = 1;
1670      1646 3      Q = CSECS_COLL_MAX+1;
1671      1647 3      WHILE .NEED GTR 0 DO
1672          BEGIN
1673              WHILE (Q=.Q-1) GEQ 0 DO IF .BV[.Q] THEN EXITLOOP;
1674                  IF (.S=.Q) LEQ 0 THEN RETURN COLL$_CMPLX;
1675                  WHILE .BV[(Q=.Q-1)] DO 0;
1676                      IF .S-.Q-1 GTR 0
1677                          THEN
1678                              BEGIN
1679                                  IF .S-.Q-1 GTR K_CHARS-1 THEN Q = .S-K_CHARS;
1680                                  NEED = NEED - (.S-.Q-1);
1681                                  IF .NEED LSS 0 THEN Q = .Q - .NEED;
1682
1683      1659 5      FOR_ALL_COLLS(P)
1684      1660 8          IF
1685      1661 8              XIF %FIELDEXPAND(COLL_ALL,2) NEQ 0
1686      1662 8                  XTHEN .P[COLL_ALL] GTR .Q AND .P[COLL_ALL] LEQ .S
1687      1663 8                  XELSE .P[COLL_C1] EQL 0 AND
1688      1664 8                      .P[COLL_C0] GTR .Q AND .P[COLL_C0] LEQ .S
1689      1665 8
1690      1666 8          XFI
1691      1667 8          THEN
1692      1668 9          BEGIN
1693      1669 9              P[COLL_C1] = .P[COLL_C0] - .Q;
1694      1670 9              P[COLL_C0] = .S;
1695      1671 8          END;
1696      1672 5          END_ALL_COLLS(P);
1697      1673 5
1698      1674 4          END;
1699      1675 3
1700      1676 3
1701      1677 3      S = COMPRESS(BV[0]);
1702      1678 3      IF_ERROR_( .S ) THEN RETURN .S;
1703      1679 3
1704      1680 3
1705      1681 2      END
1706      1682 3
1707      1683 3      ELSE BEGIN
1708      1684 3          Try converting double to single
1709      1685 3
1710      1686 3          We can convert a double collating values <x,y> to a single collating
1711      1687 3          value if either:
1712      1688 3          There are no collating values of the form: <x,0> or <z,x>, or
1713      1689 3          There are no collating values of the form: <y,0> or <y,z>.
1714      1690 3          And (additionally), of double collating values of the form: <x,z>,
1715      1691 3          <x,y> has the y with the largest (or smallest) value.
1716      1692 3
1717      1693 3
1718      1694 2          0;
1719      1695 2      END;
```

```

: 1720      1696 2    ; Check that the pad character is not the second character of a double
: 1721      1697 2    ; character.
: 1722      1698 2    ;
: 1723      1699 2    ;
: 1724      1700 2    0:
: 1725      1701 2    RETURN SSS_NORMAL;
: 1726      1702 1    END;

```

			00FC	8F	BB 00000 COMPRESS M:			
	SE	C0	AE	9E 00004	PUSHR #^M<R2,R3,R4,R5,R6,R7>			: 1525
	50	050C	CA	9E 00008	MOVAB -64(SP), SP			: 1571
	51	06	AA	3C 0000D	MOVAB 1292(R10), ST			: 1572
			10	11 00011	MOVZWL 6(CS), I			
09	AA		60	91 00013	BRB 2\$			
			1A	13 00017	CMPB (ST), 9(CS)			
09	AA	01	A0	91 00019	BEQL 3\$: 1573
			13	13 0001E	CMPB 1(ST), 9(CS)			
	50	06	C0	00020	BEQL 3\$			
	ED	51	F4	00023	ADDL2 #6, ST			: 1574
	50	09	AA	9A 00026	SOBGEQ I, 1\$: 1571
		010E	CA40	DF 0002A	MOVZBL 9(CS), R0			: 1575
			9E	B5 0002F	PUSHAL 270(CS)[R0]			
			0A	13 00031	TSTW @(SP)+			
	50	00000006	8F	D0 00033	BEQL 5\$			
			0157	31 0003A	MOVL #COLL\$_PAD, R0			
	51		6E	9E 0003D	BRW 35\$			
			FEB8	30 00040	MOVAB BV, R1			: 1580
	F4		50	E9 00043	BSBW COMPRESS			
	57	010C	CA	9E 00046	BLBC S, 4\$: 1581
	51		57	D0 0004B	MOVAB 268(R10), R7			: 1588
	53		04	D0 0004E	MOVL R7, P			
	52		01	D0 00051	MOVL #4, STEP			
	07		52	E9 00054	MOVL #1, FIRST			
	50	0100	8F	3C 00057	BLBC FIRST, 7\$			
			1A	11 0005C	MOVZWL #256, R0			
	50	06	AA	3C 0005E	BRB 11\$			
			14	11 00062	MOVZWL 6(CS), R0			: 1591
			61	B5 00064	BRB 11\$: 1589
			04	12 00066	TSTW (P)			
	08	AA	02	88 00068	BNEQ 9\$			
			A1	B5 0006C	BISB2 #2, 11(CS)			
			04	13 0006F	TSTW 2(P)			: 1590
	08	AA	04	88 00071	BEQL 10\$			
	51		53	C0 00075	B1SB2 #4, 11(CS)			
	E9		50	F4 00078	ADDL2 STEP, P			: 1591
	53		06	D0 0007B	SOBGEQ I, 8\$: 1588
	51		02	C0 0007E	MOVL #6, STEP			: 1591
	D0		52	F4 00081	ADDL2 #2, P			
	51	04	AA	3C 00084	SOBGEQ FIRST, 6\$: 1588
	51	FF00	C1	9E 00088	MOVZWL 4(CS), NEED			: 1606
05	OB	AA	02	E0 0008D	MOVAB -256(R1), NEED			
			06	AA B5 00092	BBS #2, 11(CS), 12\$			
					TSTW 6(CS)			: 1613

		02	13	00095	BEQL	13\$		
		51	D6	00097	12\$:	INCL	NEED	
		51	D5	00099	13\$:	TSTL	NEED	
		03	14	0009B		BGTR	14\$	
		00F1	31	0009D		BRW	34\$	
		52	57	000A0	14\$:	MOVL	R7, P	
		54	04	000A3		MOVL	#4, STEP	
		53	01	000A6		MOVL	#1, FIRST	
		07	53	E9	000A9	15\$:	BLBC	FIRST, 16\$
	50	0100	8F	3C	000AC		MOVZWL	#256, R0
		50	06	AA	3C	000B3	20\$	
			17	11	000B1		BRB	20\$
			02	A2	B5	000B9	16\$:	MOVZWL
				0F	13	000BC		BRB
00		55	62	3C	000BE		MOVZWL	6(CS), R0
00		6E	55	E5	000C1		(P), R5	
		02	A2	3C	000C5	18\$:	BBCC	R5, BV, 18\$
		6E	55	E5	000C9		MOVZWL	2(P), R5
		52	54	C0	000CD	19\$:	BBCC	R5, BV, 19\$
		E6	50	F4	000D0	20\$:	ADDL2	STEP, P
		54	06	DO	000D3		SOBGEQ	I, 17\$
		52	02	CO	000D6		MOVL	#6, STEP
		CD	53	F4	000D9		ADDL2	#2, P
	50	09	AA	9A	000DC		SOBGEQ	FIRST, 15\$
		010C	CA40	DF	000EO		MOVZBL	9(CS), R0
00		52	9E	3C	000E5		PUSHAL	268(CS)[R0]
		6E	52	E5	000E8		MOVZWL	a(SP)+, R2
		6E	01	8A	000EC	21\$:	BBCC	R2, BV, 21\$
		50	01	B0	000EF		BICB2	#1, BV
	53	04	AA	3C	000F2		MOVW	#1, CHAR
			53	D6	000F6		MOVZWL	4(CS), Q
			51	D5	000F8	22\$:	INCL	Q
			03	14	000FA		TSTL	NEED
		0081	31	000FC		BGTR	23\$	
			53	D7	000FF	23\$:	BRW	33\$
			04	19	00101		DECL	Q
F8		6E	53	E1	00103		BLSS	24\$
		54	53	D0	00107	24\$:	BBC	Q, BV, 23\$
		50	09	14	0010A		MOVL	Q, S
		50	8F	D0	0010C		BGTR	25\$
			7F	11	00113		MOVL	#COLL\$_CMPLX, R0
			53	D7	00115	25\$:	BRB	35\$
			53	E0	00117		DECL	Q
FA	50	6E	53	C3	0011B		BBS	Q, BV, 25\$
		54	53	C0	0011F		SUBL3	Q, S, R0
		01	50	D1	00122		CMPL	R0, #1
	00000100	8F	50	D1	00124		BLEQ	22\$
			05	15	0012B		CMPL	R0, #256
		53	FF00	C4	9E	0012D	BLEQ	26\$
		54	53	C3	00132	26\$:	MOVAB	-256(R4), Q
		51	50	C3	00136		SUBL3	Q, S, R0
	50	01	A0	9E	0013A		SUBL3	R0, NEED, R0
			03	18	0013E		MOVAB	1(R0), NEED
			53	51	C2	00140	BGEQ	27\$
			52	57	D0	00143	SUBL2	NEED, Q
			56	04	D0	00146	MOVL	R7, P
							MOVL	#4, STEP

	55	01	D0	00149	MOVL	#1, FIRST	: 1663	
	07	55	E9	0014C	BLBC	FIRST, 29\$: 1660	
	50	0100	8F	3C	0014F	MOVZWL #256, R0	
		1B	11	00154	BRB	32\$	
	50	06	AA	3C	00156	29\$: MOVZWL 6(CS), R0	1672	
		15	11	0015A	BRB	32\$	1663	
	53		62	D1	0015C	30\$: CMPL (P), Q	
		0D	15	0015F	BLEQ	31\$	
	54		62	D1	00161	CMPL (P), S	
		08	14	00164	BGTR	31\$	
02 A2	62	53	A3	00166	SUBW3	Q, (P), 2(P)	1669	
	62	54	B0	0016B	MOVW	S, (P)	1670	
	52	56	C0	0016E	ADDL2	STEP P	1672	
	E8	50	F4	00171	SOBGEQ	I, 30\$	1660	
	56	06	D0	00174	MOVL	#6, STEP	1672	
	52	02	C0	00177	ADDL2	#2, P	
	CF	55	F4	0017A	SOBGEQ	FIRST, 28\$	1660	
		FF	78	31	0017D	BRW	22\$	
	51	6E	9E	00180	33\$: MOVAB	BV, R1	1647	
		FD	75	30	00183	BSBW	COMPRESS	1677
	54		50	D0	00186	MOVL	R0, S
	05		54	E8	00189	BLBS	S, 34\$	1678
	50		54	D0	0018C	MOVL	S, R0
		03	11	0018F	BRB	35\$	
	50	01	D0	00191	34\$: MOVL	#1, R0	1701	
	5E	40	AE	9E	00194	35\$: MOVAB	64(SP), SP	1702
		00FC	8F	BA	00198	POPR	#^M<R2,R3,R4,R5,R6,R7>
				05	0019C	RSB

: Routine Size: 413 bytes, Routine Base: SOR\$RQ_CODE + 04AC

```
1728      1703 1 | Debugging routines
1729      1704 1
1730      L 1705 1 %IF %SWITCHES(DEBUG)
1731      U 1706 1 %THEN
1732      U 1707 1 LINKAGE
1733      U 1708 1 CALL = CALL;
1734      U 1709 1 %IF %BLISS(BLISS16) %THEN
1735      U 1710 1 MACRO
1736      U 1711 1     DELTA_BEGIN = DEL_BEGIN %,
1737      U 1712 1     DELTA_END = DEL_END %;
1738      U 1713 1 %FI
1739      U 1714 1 EXTERNAL ROUTINE
1740      U 1715 1     DELTA_BEGIN:    CALL,
1741      U 1716 1     DELTA:        CALL,
1742      U 1717 1     DELTA_END:     CALL,
1743      U 1718 1     SOR$OUTPUT:   CALL;
1744      U 1719 1 MACRO
1745      U 1720 1     D_(X) = UPLIT(%CHARCOUNT(X),UPLIT BYTE(X)) %
1746      U 1721 1     OUT_(X)[] = SOR$OUTPUT(D_(X) %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI) %
1747      U 1722 1
1748      U 1723 1 ROUTINE OUT_PT_1(I,CO,C1): NOVALUE =
1749      U 1724 1     OUT_('!XB(TAF)' CO=!XW, C1=!XW',
1750      U 1725 1     .I, 1, I, .CO, .C1);
1751      U 1726 1 ROUTINE OUT_PT_2: NOVALUE = OUT_(' ...');
1752      U 1727 1
1753      U 1728 1 GLOBAL ROUTINE COLL_DUMP(ADJ): CS_CALL_0 =
1754      U 1729 1
1755      U 1730 1 ++
1756      U 1731 1
1757      U 1732 1 FUNCTIONAL DESCRIPTION:
1758      U 1733 1
1759      U 1734 1 Dump the current (uncompressed) collating sequence definition.
1760      U 1735 1
1761      U 1736 1 FORMAL PARAMETERS:
1762      U 1737 1
1763      U 1738 1     ADJ      (optional) adjustment to be used when writing the "%X" form
1764      U 1739 1     of the primary table. For collating sequences with no ignored
1765      U 1740 1     or double characters, this should be specified as -1, so that
1766      U 1741 1     the dump can be used in a compilation.
1767      U 1742 1
1768      U 1743 1 IMPLICIT INPUTS:
1769      U 1744 1
1770      U 1745 1     INIT must have already been called.
1771      U 1746 1     CS is specified as a global register.
1772      U 1747 1
1773      U 1748 1 IMPLICIT OUTPUTS:
1774      U 1749 1
1775      U 1750 1     NONE
1776      U 1751 1
1777      U 1752 1 ROUTINE VALUE:
1778      U 1753 1
1779      U 1754 1     Status code
1780      U 1755 1
1781      U 1756 1 SIDE EFFECTS:
1782      U 1757 1
1783      U 1758 1     NONE
1784      U 1759 1
```

```
1785      U 1760 1 !-- BEGIN
1786      U 1761 1
1787      U 1762 1
1788      U 1763 1   CS_SETUP();
1789      U 1764 1
1790      U 1765 1   OUT_(%STRING(
1791      U 1766 1     'SIZE!=!XW, Curr_Size!=!XW, COLL_MAX!=!XW, TB!=!UB, ',
1792      U 1767 1     'DCHAR!=!XW, PAD!=!XB'),
1793      U 1768 1     .CS[CS_SIZE],
1794      U 1769 1     .CS[CS_Curr_Size],
1795      U 1770 1     .CS[CS_COLL_MAX],
1796      U 1771 1     .CS[CS_TB],
1797      U 1772 1     .CS[CS_DCHAR],
1798      U 1773 1     .CS[CS_PAD]);
1799      U 1774 1   OUT_(%STRING(
1800      U 1775 1     'MODS!=!UB, IGN!=!UB, DCOLL!=!UB!/PTAB:'),
1801      U 1776 1     .CS[CS_MODS],
1802      U 1777 1     .CS[CS_IGN],
1803      U 1778 1     .CS[CS_DCOLL]);
1804      U 1779 1   ! cs_pstatic= [$address],           ! Address of static base table
1805      U 1780 1   ! cs_ustatic= [$address],          ! Address of static upper table
1806      U 1781 1   ! cs_upper=   [$bytes(k_chars)],    ! Secondary table
1807      U 1782 1
1808      U 1783 1   DELTA_BEGIN(%B'1111',OUT_PT_1,OUT_PT_2);
1809      U 1784 1   INCR I FROM 0 TO K_CHARS=1 DO
1810      U 1785 1     BEGIN
1811      U 1786 1     LOCAL P: REF COLL_BLOCK;
1812      U 1787 1     P = CS[CS_PTAB_(.I)];
1813      U 1788 1     DELTA(.I,.P[COLL_C0], .P[COLL_C1]);
1814      U 1789 1     END;
1815      U 1790 1     DELTA_END();
1816      U 1791 1
1817      U 1792 1     OUT_('ST:');
1818      U 1793 1     FOR_ALL_DCHARS(ST)
1819      U 1794 1       OUT_('!XW(!AF) C0!=!XW, C1!=!XW',
1820      U 1795 1       .ST[ST_CHAR],
1821      U 1796 1       2, ST[ST_CHAR],
1822      U 1797 1       .BBLOCK[STEST_COL], COLL_C0],
1823      U 1798 1       .BBLOCK[STEST_COL], COLL_C1]);
1824      U 1799 1     END_ALL_DCHARS(ST);
1825      U 1800 1
1826      U 1801 1     INCR I FROM 0 TO K_CHARS/8-1 DO
1827      U 1802 1     BEGIN
1828      U 1803 1       STRUCTURE COLL_VECTOR[I] = [](COLL_VECTOR+I*COLL_K_SIZE)<0,%BPVAL,0>;
1829      U 1804 1       LOCAL P: REF COLL_VECTOR;
1830      U 1805 1       P = CS[CS_PTAB_(8*I)];
1831      U 1806 1       OUT_(%STRING(
1832      U 1807 1         'x''!XB'', x'''XB'', x'''XB'', x'''XB'', .
1833      U 1808 1         'x'''XB'', x'''XB'', x'''XB'', x'''XB'', );
1834      U 1809 1         .P[0]+.ADJ, .P[1]+.ADJ, .P[2]+.ADJ, .P[3]+.ADJ,
1835      U 1810 1         .P[4]+.ADJ, .P[5]+.ADJ, .P[6]+.ADJ, .P[7]+.ADJ);
1836      U 1811 1     END;
1837      U 1812 1
1838      U 1813 1     RETURN SSS_NORMAL;
1839      U 1814 1
1840      U 1815 1 XFI
```

```

1842 1816 1 GLOBAL ROUTINE COLLSRESULT(
1843 1817 1     COLL_SEQ:      REF VECTOR[2],          ! The collating sequence
1844 1818 1     RESLEN:       REF VECTOR[1],          ! Returned length
1845 1819 1     ) =
1846 1820 1     ++
1847 1821 1
1848 1822 1     FUNCTIONAL DESCRIPTION:
1849 1823 1
1850 1824 1         Compress the collating sequence for storage and use by the comparison
1851 1825 1         routines.
1852 1826 1
1853 1827 1     FORMAL PARAMETERS:
1854 1828 1
1855 1829 1         COLL_SEQ           a two-longword array specifying the length/address
1856 1830 1         of storage to use for the collating sequence.
1857 1831 1
1858 1832 1         RESLEN             a word (output parameter) into which the length of the
1859 1833 1         compressed collating sequence description is written.
1860 1834 1         Thus, only RESLEN bytes of the storage specified by
1861 1835 1         COLL_SEQ needs to be saved.
1862 1836 1
1863 1837 1     IMPLICIT INPUTS:
1864 1838 1
1865 1839 1         INIT must have already been called.
1866 1840 1
1867 1841 1     IMPLICIT OUTPUTS:
1868 1842 1
1869 1843 1         NONE
1870 1844 1
1871 1845 1     ROUTINE VALUE:
1872 1846 1
1873 1847 1         Status code
1874 1848 1
1875 1849 1     SIDE EFFECTS:
1876 1850 1
1877 1851 1         NONE
1878 1852 1
1879 1853 1     --
1880 1854 2     BEGIN
1881 1855 2     LOCAL
1882 1856 2         ADJ,
1883 1857 2         TAB:    XBLISS16(REF) VECTOR[K_CHARS, BYTE],
1884 1858 2         UPP:    XBLISS16(REF) VECTOR[K_CHARS, BYTE],
1885 1859 2         NEWS_P: REF VECTOR[,WORD];
1886 1860 2     MACRO
1887 M 1861 2         NEWS (X,Y) =
1888 M 1862 2         BEGIN
1889 M 1863 2         NEWS_P[0] = X; NEWS_P = NEWS_P[1];
1890 M 1864 2         %IF %NULL(Y)
1891 M 1865 2         %THEN
1892 M 1866 2         NEWS_P[0] = 0; NEWS_P = NEWS_P[1]
1893 M 1867 2         %ELSE
1894 M 1868 2         CH$WCHAR_A(.BBLOCK[Y, COLL_C0], NEWS_P);
1895 M 1869 2         CH$WCHAR_A(.BBLOCK[Y, COLL_C1], NEWS_P);
1896 M 1870 2         %FI
1897 M 1871 2         END %;
1898 M 1872 2     MACRO

```

```
: 1899      1873  2      RES_STAB_TMP = CS_UPPER %;
: 1900      1874  2
: 1901      L 1875  2
: 1902      1876  2
: 1903      1877  2
: 1904      1878  2
: 1905      1879  2
: 1906      1880  2
: 1907      1881  2
: 1908      1882  2
: 1909      U 1883  2
: 1910      U 1884  2
: 1911      U 1885  2
: 1912      U 1886  2
: 1913      U 1887  2
: 1914      U 1888  2
: 1915      U 1889  2
: 1916      U 1890  2
: 1917      1891  2
: 1918      1892  2
: 1919      1893  2
: 1920      1894  2
: 1921      1895  2
: 1922      1896  2
: 1923      1897  3
: 1924      1898  3
: 1925      1899  3
: 1926      1900  2
: 1927      1901  2
: 1928      1902  2
: 1929      1903  2
: 1930      1904  2
: 1931      1905  2
: 1932      1906  2
: 1933      1907  2
: 1934      1908  2
: 1935      1909  2
: 1936      1910  2
: 1937      L 1911  2
: 1938      U 1912  2
: 1939      U 1913  2
: 1940      1914  2
: 1941      1915  2
: 1942      1916  2
: 1943      1917  2
: 1944      U 1918  2
: 1945      U 1919  2
: 1946      U 1920  2
: 1947      U 1921  2
: 1948      U 1922  2
: 1949      U 1923  2
: 1950      1924  2
: 1951      1925  2
: 1952      1926  2
: 1953      1927  2
: 1954      1928  2
: 1955      1929  3

      XIF %BLISS(BLISS32)
      XTHEN
        EXTERNAL ROUTINE
          SOR$$COLLATE_0: ADDRESSING_MODE(LONG_RELATIVE),
          SOR$$COLLATE_1: ADDRESSING_MODE(LONG_RELATIVE),
          SOR$$COLLATE_2: ADDRESSING_MODE(LONG_RELATIVE),
          SOR$$COLLATE_0_A: ADDRESSING_MODE(LONG_RELATIVE),
          SOR$$COLLATE_1_A: ADDRESSING_MODE(LONG_RELATIVE)

      XELSE
        Because of overlay structure, Sort-11 has to resolve the
        addresses on the fly

        BIND
          SOR$$COLLATE_0 = 0,
          SOR$$COLLATE_1 = 1,
          SOR$$COLLATE_2 = 2

      XFI;

      CS_SETUP(COLL_SEQ);

      ! Compress the tables

      BEGIN LOCAL STATUS;
      STATUS = COMPRESS M();
      IF ERROR_( .STATUS ) THEN RETURN .STATUS;
      END;

      ! Compute the adjustment
      ! This is 1, unless we have: double characters or double collating values
      ! or ignored characters, in which case it is zero.
      ! If the adjustment is zero, we will use a zero in the primary table to
      ! indicate that the secondary table must be used.

      ADJ = 1;
      IF .CS[CS_DCOLL] OR .CS[CS_IGN] OR .CS[CS_DCHAR] GTR 0 THEN ADJ = 0;

      L 1911  2
      XIF %SWITCHES(DEBUG)
      XTHEN
        COLL_DUMP(-.ADJ);
      XFI

      ! Allocate the TAB and UPP tables in the work area

      XIF %BLISS(BLISS16) XTHEN
      TAB = .CS[CS_CURR_SIZE] + 2 * %SIZE(VECTOR[K_CHARS, BYTE]);
      IF .TAB GTRU .CS[CS_SIZE] THEN RETURN COLLS_CMPLX;
      CS[CS_CURR_SIZE] = .TAB;
      TAB = .TAB + CS[BASE] - 2 * %SIZE(VECTOR[K_CHARS, BYTE]);
      UPP = .TAB + %SIZE(VECTOR[K_CHARS, BYTE]);
      XFI

      ! First, compute the primary table (into tab)
      CH$FILL(0, K_CHARS, TAB[0]);
      BEGIN
```

```

: 1956      1930  3 LOCAL P: REF COLL_BLOCK;
: 1957      1931  3 P = CS[CS_PTAB] + (K_CHARS-1) * COLL_K_SIZE;
: 1958      1932  3 DECR I FROM K_CHARS-1 TO 0 DO
: 1959      1933  4 BEGIN
: 1960      1934  4   IF .P[COLL_C1] EQ 0 THEN TAB[I] = .P[COLL_C0] - .ADJ;
: 1961      1935  4   P = .P - COLL_K_SIZE;
: 1962      1936  3 END;
: 1963      1937  2
: 1964      1938  4
: 1965      1939  4 FOR_ALL_DCHARS(ST)
: 1966      1940  2   TAB[ST[CHAR_0]] = 0;
: 1967      1941  2 END_ALL_DCHARS(ST);

: 1968      1942  2
: 1969      1943  2 ! Copy the upper table
: 1970      1944  2 CHSMOVE(K_CHARS, CS[CS_UPPER], UPP[0]);
: 1971      1945  2
: 1972      1946  2 ! Don't bother using silly upper tables.
: 1973      1947  2
: 1974      1948  2
: 1975      1949  2 IF CHSEQ(0, UPP[0], K_CHARS, UPP[0], .UPP[0])
: 1976      1950  2 THEN
: 1977      1951  2   CS[CS_TB] = .CS[CS_TB] OR TB$NOUPPER;

: 1978      1952  2
: 1979      1953  2
: 1980      1954  2 ! Order the entries in the cs_stab table by the character codes.
: 1981      1955  2 ! This is needed if there are several double characters with the
: 1982      1956  2 same first character. Note that the entry with the smallest value
: 1983      1957  2 must be the first one accessed by the for_all_dchars macro.
: 1984      1958  2 ! This code depends on the for_all_dchars macro accessing the entries
: 1985      1959  2 in order from lower addresses to higher addresses.
: 1986      1960  2
: 1987      1961  3 BEGIN
: 1988      1962  3 MACRO
: 1989      1963  3   SWAP_(X,Y) = (T = .X; X = .Y; Y = .T) %,
: 1990      M 1964  3   SWAP_ST(X,Y) =
: 1991      M 1965  3   BEGIN
: 1992      M 1966  3   LOCAL T.
: 1993      M 1967  3   SWAP_(X[ST_CHAR], Y[ST_CHAR]);
: 1994      M 1968  3   XIF %FIELDEXPAND(ST_COLL,2) NEQ 0
: 1995      M 1969  3   XTHEN
: 1996      M 1970  3   SWAP_(X[ST_COLL], Y[ST_COLL]);
: 1997      M 1971  3 XELSE
: 1998      M 1972  3   SWAP_(BBLOCK[X[ST_COLL],COLL_C0],BBLOCK[Y[ST_COLL],COLL_C0]);
: 1999      M 1973  3   SWAP_(BBLOCK[X[ST_COLL],COLL_C1],BBLOCK[Y[ST_COLL],COLL_C1]);
: 2000      M 1974  3 XFI
: 2001      1975  3 END %;
: 2002      1976  3 LOCAL
: 2003      1977  3   ST_MIN: REF ST_BLOCK,
: 2004      1978  3   ST_1:  REF ST_BLOCK,
: 2005      1979  3   ST_2:  REF ST_BLOCK;
: 2006      1980  3   ST_1 = CS[CS_STAB];
: 2007      1981  3   DECR I FROM :CS[CS_DCHAR]-1 TO 1 DO
: 2008      1982  4   BEGIN
: 2009      1983  4   ST_MIN = ST_1[BASE];
: 2010      1984  4   ST_2 = ST_1[BASE];
: 2011      1985  4   DECR J FROM .I-1 TO 0 DO
: 2012      1986  5   BEGIN

```

: 2013 1987 5 ST_2 = ST_2 + ST_K_SIZE;
: 2014 1988 5 IF .ST_2[ST_CHAR] < ST_MIN[ST_CHAR] THEN ST_MIN = ST_2[BASE_];
: 2015 1989 4 END;
: 2016 1990 4 SWAP_ST(ST_MIN, ST_1);
: 2017 1991 4 ST_1 = ST_T + ST_K_SIZE;
: 2018 1992 3 END;
: 2019 1993 2 END;
: 2020 1994 2 ! Now compute the secondary table
: 2021 1995 2 ! We compute it to cover the cs_upper table.
: 2022 1996 2 ! It may extend far enough to cover the cs_ptab table, but we should
: 2023 1997 2 always be ahead, unless there are more than k_chars double characters.
: 2024 1998 2 NEWS_P = CS[RES_STAB_TMP];
: 2025 1999 2 IF .ADJ EQL 0
: 2026 2000 2 THEN
: 2027 2001 2 BEGIN
: 2028 2002 2 ! This must be an incr loop
: 2029 2003 2 INCR PT_IDX FROM 0 TO K_CHARS-1 DO IF .TAB[PT_IDX] EQL 0 THEN
: 2030 2004 3 BEGIN
: 2031 2005 3 LOCAL
: 2032 2006 3 ENTRY;
: 2033 2007 3 ENTRY = FALSE;
: 2034 2008 3 IF
: 2035 2009 4 XIF %FIELDEXPAND(COLL_ALL,2) NEQ 0
: 2036 2010 4 XTHEN .CS[CS_PTAB_(.PT_IDX)] NEQ 0
: 2037 2011 4 XELSE
: 2038 2012 4 BEGIN
: 2039 2013 4 LOCAL P: REF COLL_BLOCK;
L 2040 2014 4 P = CS[CS_PTAB_(.PT_IDX)];
: 2041 2015 4 .P[COLL_C0] NEQ 0 OR .P[COLL_C1] NEQ 0
: 2042 U 2016 4 END
: 2043 U 2017 4 XFI
: 2044 U 2018 4 THEN
: 2045 U 2019 4 BEGIN
: 2046 U 2020 4 NEWS_(%X'FF00'+.PT_IDX, CS[CS_PTAB_(.PT_IDX)]);
: 2047 U 2021 4 ENTRY = TRUE;
: 2048 2022 4 END;
: 2049 2023 4 FOR_ALL_DCHARS(ST)
: 2050 2024 5 IF .ST[ST_CHAR_0] EQL .PT_IDX<0,8,0>
: 2051 2025 5 THEN
: 2052 2026 5 BEGIN
: 2053 2027 4 IF NOT .ENTRY
: 2054 2028 6 THEN
: 2055 2029 6 BEGIN
: 2056 2030 6 NEWS_(%X'FF00'+.PT_IDX, CS[CS_PTAB_(.PT_IDX)]);
: 2057 2031 7 ENTRY = TRUE;
: 2058 2032 7 NEWS_(.ST[ST_CHAR], ST[ST_COLL]);
: 2059 2033 7 END;
: 2060 2034 7 END_ALL_DCHARS(ST);
: 2061 2035 7 END;
: 2062 2036 7 NEWS_(%X'FFFF');
: 2063 2037 6 NEWS_(%X'0000');
: 2064 2038 4 END;
: 2065 2039 3 END;
: 2066 2040 3 END;
: 2067 2041 3 END;
: 2068 2042 2 END;
: 2069 2043 2 END;

```

: 2070      2044  2
: 2071      2045  2
: 2072      2046  2
: 2073      2047  3
: 2074      2048  3
: 2075      2049  3
: 2076      2050  3
: 2077      2051  3
: 2078      2052  3
: 2079      2053  3
: 2080      2054  3
: 2081      2055  3
: 2082      2056  3
: 2083      2057  3
: 2084      2058  3
: 2085      2059  3
: 2086      2060  3
: 2087      2061  4
: 2088      2062  4
: 2089      2063  3
: 2090      2064  3
: 2091      2065  3
: 2092      2066  4
: 2093      2067  4
: 2094      2068  3
: 2095      2069  3
: 2096      2070  3
: 2097      2071  3
: 2098      2072  3
: 2099      2073  3
: 2100      2074  3
: 2101      2075  3
: 2102      2076  2
: 2103      2077  2
: 2104      2078  2
: 2105      2079  1

        ! Store the tables, values, and the address of the routine to use
        BEGIN LOCAL
          TMP
          SAVE: VECTOR[3,BYTE]:
          TMP = NEWS P[0] - CS[RES_STAB TMP];
          RESLEN[0] = %FIELDEXPANDRES STAB,0) + .TMP;
          IF .RESLEN[0] GTRU .CS[CS_SIZE]
            %BLISS16( OR CS[RES_STAB_TMP]+.TMP GTRA TAB[0])
        THEN
          RETURN COLL$[Cmplx];
          SAVE[0] = .CS[CS_TB];
          SAVE[1] = .CS[CS_PAD];
          SAVE[2] = .CS[CS_REVERSE];
          CS[RES RTN] =
            (IF .ADJ NEQ 0 THEN SOR$COLLATE 0
             ELIF .CS[CS_DCHAR] EQL 0 THEN SOR$COLLATE_1
             ELSE SOR$COLLATE 2);
        %IF %BLISS(BLISS32) %THEN
          CS[RES RTN_A] =
            (IF .ADJ NEQ 0 THEN SOR$COLLATE 0_A
             ELIF .CS[CS_DCHAR] EQL 0 THEN SOR$COLLATE_1_A
             ELSE 0);
        %FI
        CH$MOVE(.TMP, CS[RES_STAB TMP], CS[RES_STAB]);
        CH$MOVE(K_CHARS, UPP[0], CS[RES_UPPER]);
        CH$MOVE(K_CHARS, TAB[0], CS[RES_PTAB]);
        CS[RES_TB] = .SAVE[0];
        CS[RES_PAD] = .SAVE[1];
        CS[RES_REVERSE] = .SAVE[2];
      END;

      RETURN SSS_NORMAL;
    END;
  
```

			.EXTRN SOR\$COLLATE_0, SOR\$COLLATE_1	
			.EXTRN SOR\$COLLATE_2, SOR\$COLLATE_0_A	
			.EXTRN SOR\$COLLATE_1_A	
		OFFC 00000	.ENTRY COLL\$RESULT, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 1816	
	5E	FDFC	CE 9E 00002	MOVAB -516(SP), SP
	50	04	AC D0 00007	MOVL COLL SEQ, R0
	5A	04	A0 D0 0000B	MOVL 4(R0), CS
			BSBW COMPRÉSS M	
		FE51	BLBS STATUS, TS	
		01	RET	
			MOVL #1, ADJ	
	0A	59	01 D0 00016 1\$: BBS #2, 11(CS), 2\$	
05	0B	AA	02 E0 00019	
	0B	AA	01 E0 0001E	
		06	BBS #1, 11(CS), 2\$	
		AA	TSTW 6(CS)	
		B5 00023	BEQL 3\$	
		02 13 00026	CLRL ADJ	
		59 D4 00028 2\$:		

0100	8F	00	6E	00	2C	0002A	3\$:	MOVCS	#0, (SP), #0, #256, TAB	: 1928	
			FF00	CD	00031						
			0508	CA	9E	00034		MOVAB	1288(R10), P	: 1931	
			FF	8F	9A	00039		MOVZBL	#255, I	: 1932	
			02	A1	B5	0003D	4\$:	TSTW	2(P)	: 1934	
				07	12	00040		BNEQ	5\$		
				59	83	00042		SUBB3	ADJ, (P), TAB[1]		
				04	C2	00049	5\$:	SUBL2	#4, P		
			EE	50	F4	0004C		SOBGEQ	I, 4\$		
			57	CA	9E	0004F		MOVAB	1292(R10), R7		
			52	57	D0	00054		MOVL	R7, ST		
			51	06	AA	3C	00057		MOVZWL	6(CS), I	
				08	11	0005B		BRB	7\$		
			50	82	9A	0005D	6\$:	MOVZBL	(ST)+, R0	: 1939	
			FF00	CD40	94	00060		CLRB	TAB[R0]		
			52	05	C0	00065		ADDL2	#5, ST		
			F2	51	F4	00068	7\$::	SOBGEQ	I, 6\$		
			58	0C	AA	9E	0006B	MOVAB	12(CS), R8		
			68	0100	8F	28	0006F	MOV C3	#256, (R8), UPP		
				00	2D	00076		CMP C5	#0, UPP, UPP, #256, UPP		
			04	AE		0007F				: 1949	
				04	12	00081		BNEQ	8\$		
			08	AA	02	88	00083	BISB2	#2, 8(CS)		
				52	57	D0	00087	8\$::	MOVL	R7, ST_1	
				51	06	AA	3C	0008A	MOVZWL	6(CS), I	
					2D	11	0008E	BRB	12\$		
			53	52	D0	00090	9\$::	MOVL	ST_1, ST_MIN		
			54	52	D0	00093		MOVL	ST_1, ST_2		
			50	51	D0	00096		MOVL	I, J		
				08	11	00099		BRB	11\$		
			54	06	C0	0009B	10\$::	ADDL2	#6, ST_2		
			63	64	B1	0009E		CMPW	(ST_2), (ST_MIN)		
				03	1E	000A1		BGEQU	11\$		
			53	54	D0	000A3		MOVL	ST_2, ST_MIN		
			F2	50	F4	000A6	11\$::	SOBGEQ	J, 10\$		
			50	63	3C	000A9		MOVZWL	(ST_MIN), T		
			63	62	B0	000AC		MOVV	(ST_1), (ST_MIN)		
			82	50	B0	000AF		MOVV	T, (ST_1)+		
			50	A3	D0	000B2		MOVL	2(ST_MIN), T		
			82	62	D0	000B6		MOVL	(ST_T), 2(ST_MIN)		
			D0	50	D0	J00BA	12\$::	MOVL	T, (ST_1)+		
				51	F5	000BD		SOBGTR	I, 9\$		
				50	D0	000C0		MOVL	R8, NEWS_P		
				59	D5	000C3		TSTL	ADJ		
				69	12	000C5		BNEQ	20\$		
				51	D4	000C7		CLRL	PT_IDX		
			FF00	CD41	95	000C9	13\$::	TSTB	TAB[PT_IDX]		
				51	12	000CE		BNEQ	19\$		
				55	D4	000D0		CLRL	ENTRY		
			53	010C	CA41	DE	000D2	MOVAL	268(CS)[PT_IDX], R3		
				63	D5	000D8		TSTL	(R3)		
				10	13	000DA		BEQL	14\$		
			80	FF00	8F	A1	000DC	ADDW3	#65280, PT_IDX, (NEWS_P)+		
				63	90	000E2		MOVB	(R3), (NEWS_P)+		
				02	A3	90	000E5	MOVB	2(R3), (NEWS_P)+		
			80	55	01	D0	000E9	MOVL	#1, ENTRY		
				54	57	D0	000EC	14\$::	MOVL	R7, ST	

			56	06	AA	3C 000EF		MOVZWL	6(CS), I	: 2029
			29	11	000F3			BRB	18\$	
			51	64	91	000F5	15\$:	CMPB	(ST), PT_IDX	
			21	12	000F8			BNEQ	17\$	
			80	0D	55	E8 000FA		BLBS	ENTRY, 16\$	
			51	FF00	8F	A1 000FD		ADDW3	#65280, PT_IDX, (NEWS_P)+	2032
			80		63	90 00103		MOV8	(R3), (NEWS_P)+	2034
			80	02	A3	90 00106		MOV8	2(R3), (NEWS_P)+	
			55		01	DO 0010A	16\$:	MOVL	#1, ENTRY	
			80		64	BO 0010D		MOVW	(S1), (NEWS_P)+	
			52	02	A4	9E 00110		MOVAB	2(ST), R2	
			80		62	90 00114		MOV8	(R2), (NEWS_P)+	
			80	02	A2	90 00117		MOV8	2(R2), (NEWS_P)+	
			54		06	CO 0011B	17\$:	ADDL2	#6, S1	
			D4		56	F4 0011E	18\$:	SOBGEQ	I, 15\$	
			A0	51	000000FF	8F F3 00121	19\$:	AOBLEQ	#255, PT_IDX, 13\$	
			80		FFF	8F 3C 00129		MOVZWL	#65535, ?NEWS_P)+	
						80 D4 0012E		CLRL	(NEWS_P)+	
			52	50	58	C3 00130	20\$:	SUBL3	R8, NEWS_P, TMP	
08 BC		08	BC	020C	C2	9E 00134		MOVAB	524(R2), @RESLEN	
			6A	10	00	ED 0013A		CMPZV	#0, #16, (CS), @RESLEN	
					08	1E 00140		BGEQU	21\$	
				50 00000000G	8F	DO 00142		MOVL	#COLL\$_CMPLX, R0	
						04 00149		RET		2056
			02	6E	08	AA 80 0014A	21\$:	MOVW	8(CS), SAVE	
				AE	0A	AA 90 0014E		MOV8	10(CS), SAVE+2	
						51 D4 00153		CLRL	R1	
						59 D5 00155		TSTL	ADJ	
						08 13 00157		BEQL	22\$	
						51 D6 00159		INCL	R1	
				50 00000000G	EF	9E 0015B		MOVAB	SOR\$\$COLLATE_0, R0	
					15	11 00162		BRB	24\$	
					06	AA B5 00164	22\$:	TSTW	6(CS)	
					09	12 00167		BNEQ	23\$	
				50 00000000G	EF	9E 00169		MOVAB	SOR\$\$COLLATE_1, R0	
					07	11 00170		BRB	24\$	
				50 00000000G	EF	9E 00172	23\$:	MOVAB	SOR\$\$COLLATE_2, R0	
				6A	50	DO 00179	24\$:	MOVL	R0, (CS)	
				09	51	E9 0017C		BLBC	R1, 25\$	
				50 00000000G	EF	9E 0017F		MOVAB	SOR\$\$COLLATE_0_A, R0	
					10	11 00186		BRB	27\$	
					06	AA B5 00188	25\$:	TSTW	6(CS)	
					09	12 0018B		BNEQ	26\$	
				50 00000000G	EF	9E 0018D		MOVAB	SOR\$\$COLLATE_1_A, R0	
					02	11 00194		BRB	27\$	
					50	D4 00196	26\$:	CLRL	R0	
					DO	00198	27\$:	MOVL	R0, 4(CS)	
			020C	CA	04	AA		MOV C3	TMP, (R8), 524(CS)	
			010C	CA	68	68		MOV C3	#256, UPP, 268(CS)	
					04	AE	0100	MOV C3	#256, TAB, (R8)	
				68	CD	0100	8F 28 001A2	MOVW	SAVE, 8(CS)	
					08	AA	28 001AB	MOV8	SAVE+2, 10(CS)	
					0A	AA	02 AE 90 001B7	MOVL	#1, R0	
					50	01	DO 001BC	RET		
						04 001BF				

: Routine Size: 448 bytes, Routine Base: SOR\$RO_CODE + 0649

```
: 2106      2080 1
: 2107      2081 1 END
: 2108      2082 0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	2057 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)	
. ABS .	0 NOVEC,NOWRT,NORD ,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0)	

Library Statistics

File	Total	Symbols	Percent	Pages Mapped	Processing Time
_S255\$DUA28:[SYSLIB]XPORT.L32:1	590	36	6	252	00:00.1

COMMAND QUALIFIERS

```
BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:SORCOLUTI/OBJ=OBJ$:$SORCOLUTI MSRC$:$SORCOLUTI/UPDATE=(ENH$:$SORCOLUTI
)
```

```
: Size: 2057 code + 0 data bytes
: Run Time: 00:53.9
: Elapsed Time: 02:44.7
: Lines/CPU Min: 2317
: Lexemes/CPU-Min: 38807
: Memory Used: 244 pages
: Compilation Complete
```

0363 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SORCOMMAND
LIS

LIBFIXUPD
LIS

REQSYM
R32

CRETRANS
LIS

SFKEYWORD
LIS

OPCODES
LIS

SORCOLLAT
LIS

SORARCHAT
LIS

SORCOLUTI
LIS